Recovering a code’s length and synchronization from a noisy intercepted bitstream.

M. Cluzeau and M. Finiasz
We intercept a noisy bitstream and want to recover the (encrypted) information.
Overview of the problem

Most of the time, coding schemes are standardized
▶ no need for code reconstruction.

Yet, “some people” are interested in this:
▶ not many public works on this topic,
▶ many interesting problems arise, depending on the type of code we focus on.

Here we focus on linear block codes requiring to:
▶ find the block length,
▶ find a generator/parity check matrix,
▶ find an efficient decoder,
→ we do not address this problem here.
The only thing we have is a noisy bitstream:

- we need to find $s_0$ and $n_0$ the synchronization and length of the code.

For very short codes of small dimension various techniques can give us some hint on $n$,

- none of them work for real life codes...
  - we have to test each choice of $s$ and $n$. 

The case of linear block codes

\[
\begin{array}{c@{}c@{}c@{}c@{}c@{}c@{}c@{}c@{}c@{}c@{}c@{}c}
010010110100001011101010010100101110...
\end{array}
\]
In the absence of noise
The easy case...

For given $s$ and $n$ build the matrix $\mathcal{G}$ of “codewords”

- if $n = n_0$ and $s = s_0$ it has minimal rank $k$,
- if $n = n_0$ and $s \neq s_0$ it has rank $\min(k + |s - s_0|, n)$,
- if $n \neq n_0$ it has rank $n$. 
For given \( s \) and \( n \) build the matrix \( \mathcal{G} \) of “codewords”

- if \( n = n_0 \) and \( s = s_0 \) it has minimal rank \( k \),
- if \( n = n_0 \) and \( s \neq s_0 \) it has rank \( \min(k + |s - s_0|, n) \),
- if \( n \neq n_0 \) it has rank \( n \).

Very efficient to guess \( n_0 \) and then \( s_0 \),
- only for very low noise levels \( \tau \ll \frac{1}{n} \).

For higher noises the rank is always \( n \)...
In the presence of noise
Using words of the dual

If $n$ and $s$ are correct, a word of the dual of the target code multiplied by $G$ should have low weight,

suppose we have such a dual word of low weight $w$. 

![Diagram]

$1 - (1 - 2\tau)^w$
In the presence of noise
Using words of the dual

If a word following the green distribution is found, \( n = n_0 \)

and \( s - s_0 \) is probably small.
The algorithm we propose

▶ We need to exhaustively search through the possible $s$ and $n$.

▶ Successively go through the possible values of $n$
  ▶ for each length “test” several synchronizations $s$
    → different possible heuristics.

▶ Testing a pair $(n, s)$ consists in searching for a dual word following the green distribution:
  ▶ exhaustive search of words of given weight
  ▶ using Valembois’ algorithm.
Exhaustive search of given weight dual words

We look for a dual word of length $n$ and weight $w$.

We can find all such dual words using:

- straight-forward exhaustive search
  - $O(n^w)$ time and $O(1)$ memory.
- the birthday algorithm
  - $O(n^{w/2})$ time and $O(n^{w/2})$ memory.
- the Chose-Joux-Mitton algorithm [Eurocrypt 2002]
  - $O(n^{w/2})$ time and $O(n^{\lceil w/4 \rceil})$ memory.

Very efficient for codes with very low weight dual words typically LDPC codes.
Valembois’ algorithm

- Based on the Canteaut-Chabaud decoding algorithm,
- does not focus only on low weight dual words,
- small memory requirements.

▷ Very efficient for low noise levels,
  → tolerates higher noise levels for very short codes.
Practical results
Random linear codes

▶ Codes of rate $\frac{1}{2}$:

▷ no low weight dual words,
▷ for our problem: among the difficult cases.

▶ Dual words found in 10000 iterations of Valembois' algorithm (less than a second).

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\tau$</th>
<th>0.001</th>
<th>0.002</th>
<th>0.005</th>
<th>0.01</th>
<th>0.02</th>
<th>0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td></td>
<td>14637</td>
<td>27081</td>
<td>42570</td>
<td>42913</td>
<td>19464</td>
<td>210</td>
</tr>
<tr>
<td>64</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>1172189</td>
<td>6310</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>2992</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Practical results

LDPC codes

- LDPC codes of rate $\frac{1}{2}$ and weight 6 parity checks,
  - find words for lengths up to 10000 with 2GB memory.

- For an LDPC of length 1000 in 50 iterations ($\sim$2 min.)

<table>
<thead>
<tr>
<th>$\tau$</th>
<th>words found</th>
<th>expected words per iteration</th>
<th>expected total words found</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>478</td>
<td>41</td>
<td>492</td>
</tr>
<tr>
<td>0.02</td>
<td>251</td>
<td>7.5</td>
<td>266</td>
</tr>
<tr>
<td>0.03</td>
<td>84</td>
<td>1.5</td>
<td>70</td>
</tr>
<tr>
<td>0.04</td>
<td>15</td>
<td>0.33</td>
<td>16</td>
</tr>
<tr>
<td>0.05</td>
<td>6</td>
<td>0.08</td>
<td>3.9</td>
</tr>
<tr>
<td>0.06</td>
<td>1</td>
<td>0.02</td>
<td>1.0</td>
</tr>
</tbody>
</table>
We can find the length/synchronization of a code by using reconstruction techniques, easier for codes with low weight dual words → LDPC, not very satisfying for random codes.

For an unknown code, both techniques should be tried for very low noise levels, Valembois’ algorithm is faster, even for long LDPC codes.

For other kind of codes:
- convolutional codes [Côte, Sendrier - ISIT09]
- turbocodes → we are working on it...