Abstract (read only this if you are lazy)

The \( l \)-th stopping redundancy \( \rho(C) \) of the binary \([n, k, d]\) code \( C \), \( 1 \leq l \leq d \), is defined as the minimum number of rows in the parity-check matrix of \( C \), such that the smallest stopping set is of size at least \( l \). The stopping redundancy \( \rho(C) \) is defined as \( \rho_2(C) \). In this work, we improve on the probabilistic analysis of stopping redundancy, proposed by Han, Siegel and Vardy, which yields the best bounds known today. In our approach, we judiciously select the first few rows in the parity-check matrix, and then continue with the probabilistic method. By using similar techniques, we improve also on the best known bounds on \( \rho(C) \), for \( 1 \leq l \leq d \). Our approach is compared to the existing methods by numerical computations.

**Motivation: MP decoding failures**

Settings:
- \( C \subset F_2^n \) is a linear \([n, k, d]\) code, \( C^\perp \) is dual to \( C \)
- BEC, Message-passing (MP) decoding

Some facts:
- Decoder fails \( \Rightarrow \) stopping set is erased
- Stopping sets undesirable
- Erasure of stopping sets of small size is more probable
- \( \rho_1 \) Stopping sets are defined for \( H \), not for \( C \)
- Additional (redundant) rows could eliminate stopping sets
- Every codeword of \( C \) induces stopping set \( \Rightarrow \) those not possible to eliminate

Idea: use redundant \( H \) which eliminates all stopping sets of size \( \leq d \)

- Stopping redundancy \( \rho(C) \) is the minimum number of rows in \( H \) s.t. there are no stopping sets of size \( \leq d \)
- Always achievable: \( \rho(C) \leq 2^{n-k} - 1 \)

**Stopping sets: example**

\( C = [10, 3, 4] \) with parity-check matrix \( H \):

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

Stopping sets of size less than 4: \{1, 3, 10\}, \{1, 5, 8\}, \{4, 8, 10\}. (bold and blue – examples)

We could add redundant rows to this matrix in order to eliminate these stopping sets.

Adding redundant parity checks corresponds to introducing check nodes:

**Stops during decoding**

\( C \subset F_2^n \) is a linear \([n, k, d]\) code, \( C^\perp \) is dual to \( C \)

- Decoder fails
- \( \Rightarrow \) stopping set is erased

**Probabilistic approach (Han-Siegel-Vardy’08)**

**Step 1**
1. Choose \( t \) random codewords of \( C^\perp \) (no repetition)
2. Find expectations of:
   - Number of stopping sets left
   - Decrease of number of stopping sets left
3. Guaranteed existence

**Step 2**
1. Add one more random codeword of \( C^\perp \) (no repetition)
2. Find expectations of:
   - Decrease of number of stopping sets left
   - Increase of rank
3. Guaranteed existence

Erase.

Stop when there are no small stopping sets left and rank is \( n - k \)

**Our improvements**

- **Main trick:**
  - Choose some first row(s) non-randomly and carefully
  - ... so that we know how many stopping sets left
  - ... or can bound their number
  - And then apply technique of Han-Siegel-Vardy

Candidates for non-random rows:
- One or couple of rows of weight \( d^\perp \)
- Just take some codewords of \( C^\perp \), calculate straightforward (might be slow and not optimal)

**Numerical results**

We compare the bounds on the stopping redundancy obtained in [2] and [1] with our results. We consider two codes: the extended \([24, 12, 8]\) binary Golay code and the extended \([48, 24, 12]\) binary Quadratic Residue (QR) code. Both of them are known to be self-dual.

**Upper bounds on the stopping redundancy**

<table>
<thead>
<tr>
<th>Code</th>
<th>24, 12, 8 Golay</th>
<th>48, 24, 12 QR</th>
</tr>
</thead>
<tbody>
<tr>
<td>2, Thm 4</td>
<td>2509</td>
<td>4510385</td>
</tr>
<tr>
<td>1, Thm 1</td>
<td>198</td>
<td>3655</td>
</tr>
<tr>
<td>1, Thm 3</td>
<td>194</td>
<td>3655</td>
</tr>
<tr>
<td>1, Thm 4</td>
<td>187</td>
<td>3577</td>
</tr>
<tr>
<td>1, Thm 7</td>
<td>182</td>
<td>3564</td>
</tr>
<tr>
<td>One row of weight ( d^\perp )</td>
<td>180</td>
<td>3638</td>
</tr>
<tr>
<td>Two rows of weight ( d^\perp )</td>
<td>177</td>
<td>3515</td>
</tr>
</tbody>
</table>

**References**


Yauhen Yakimenka, Vitaly Skachek

Institute of Computer Science, University of Tartu, Estonia

yauhen@ut.ee, vitaly@ut.ee