On Optimum Decoding of Certain Product Codes

Enrico Paolini
University of Bologna, Italy

work with: Gianluigi Liva and Marco Chiani

Munich Workshop on Coding and Modulation
Munich, Germany, July 31st, 2015
## WS Topics

- Non-binary LDPC and Turbo codes
- Spatially-coupled codes
- Polar codes
- Lattice codes
- Decoding for short block lengths
Outline

- ML decoding of product codes
- Reduced complexity decoder
- Numerical results
- Conclusion
Product Codes

- Product code construction introduced in [Elias1954].
- Recent literature addresses efficient iterative decoding (e.g., [Tanner1981]–[Pyndiah1998]).
- [Wolf1978]: maximum-likelihood (ML) decoding can be performed very efficiently for some product codes.

---

Product Code Trellis

- [Wolf1978]: Trellis representation with a maximum number of states

\[
\min\{2^{(n_1-k_1)k_2}, 2^{(n_2-k_2)k_1}\}
\]

per trellis section.

- Particularly beneficial when one component code has low dimension (e.g., small \(k_2\)) and one has a small number of parity bits (e.g., small \(n_1 - k_1\)).

- Particular case: the high rate code is a single parity-check code \((n_1 - k_1 = 1)\). In this case:

\[2^{k_2}\] states per trellis section.

- Hereafter we focus on this class of product codes. Row code \(C_1\) is SPC, column code \(C_2\) is any linear block code.
Some Notation (1/2)

\[ k_1 \]

\[ k_2 \]

\[ \text{U} \]

\[ C_1 \rightarrow \]

\[ n_1 = k_1 + 1 \]

\[ \text{M} : [m_1^T, m_2^T, \ldots, m_{n_1}^T] \]

\[ m_1^T + m_2^T + \cdots + m_{n_1}^T = 0^T \]

\[ c = [c_1, c_2, \ldots, c_{n_1}] \]

\[ x = [x_1, x_2, \ldots, x_{n_1}] \]

\[ x_i = 1 - 2c_i \]

\[ y = [y_1, y_2, \ldots, y_{n_1}] \]

\[ y_i = x_i + \nu_i \]

\[ n_1 = k_1 + 1 \]

\[ \text{C} : [c_1^T, c_2^T, \ldots, c_{n_1}^T] \]

\[ n_2 - k_2 \]
Some Notation (2/2)

- Length-$k_2$ vectors $m_1^T, m_2^T, \ldots, m_{n_1}^T$ regarded as the binary vector representations of the elements of finite field $\mathbb{F}_q$ with $q = 2^{k_2}$.

- $[\mu_1, \mu_2, \ldots, \mu_{n_1}] := [m_1^T, m_2^T, \ldots, m_{n_1}^T]$, $\mu_i \in \mathbb{F}_q$, thus
  \[ \mu_1 + \mu_2 + \cdots + \mu_{n_1} = 0 \]

- Encoder (equivalent perspective):

  \[
  (n_1, n_1 - 1) \text{ SPC over } \mathbb{F}_q \\
  C_q \xrightarrow{\mathbf{c}' = (\mu_1, \mu_2, \ldots, \mu_{n_1})} \text{ modulator} \xrightarrow{\mathbf{M}} \mathbf{x} = [x_1 \ x_2 \ \cdots \ x_{n_1}] \\
  \mathbf{H} = [1 \ 1 \ \cdots \ 1] \\
  \mathbf{M} : \mu_i \in \mathbb{F}_q \mapsto x_i = 1 - 2c_i
  \]
ML Decoding

- SPC code $C_q$ Viterbi decoded over its trellis.
- The trellis comprises $q = 2^{k_2}$ states (apart from terminations). States are of subsequent layers are “fully connected”. Example ($k_2 = 2$):

  $S_{i-1}$

  ![Trellis Diagram]

  $S_i$

  - Branch metrics $\langle x_i, y_i \rangle$.
  - Label of edge from $S_{i-1} = s$ to $S_i = s'$: $s + s' \in \mathbb{F}_q$.

- Neglecting terminations, complexity proportional to $k_1 q^2 = k_1 2^{2k_2}$. 

- Reduced Complexity Decoder
- Numerical Results
- Conclusion
Example

- Row code: (8, 7) SPC code. Column code: (24, 12) Golay code.
- $n = 192$, $k = 84$, $d = 16$, $A_{\text{min}} = 28 \times 759 = 21252$.
- Number of states: $2^{12}$. Number of edges per trellis section: $2^{24}$.

Q: Possible to reduce complexity while preserving performance?
Symbol-Wise MAP Decoding

- To reduce complexity:
  - Switch to symbol-wise optimum MAP decoding:
    
    $$\hat{\mu}_i = \arg \max_{\omega \in \mathbb{F}_q} \Pr\{\mu_i = \omega | y\}$$

  - Use fast Fourier transform.

- Using BCJR we have
  
  $$L_i(\omega) := \Pr\{\mu_i = \omega | y\} = \sum_{s, s': s + s' = \omega} \varphi_{i-1}(s)\gamma_i(s, s')\beta_i(s')$$

  with standard meaning for the forward ($\varphi$), backward ($\beta$), and branch transition ($\gamma$) metrics.
Defining $s' = s + \omega$, the branch transition metric may be computed as

$$
\gamma_i(s, s') \propto (2\pi\sigma^2)^{n_2/2} \exp \left(-\frac{1}{2\sigma^2} \langle y_i, M(\omega) \rangle \right)
$$

$$=: \gamma_i(s + s')$$

$$=: \gamma_i(\omega)$$

- An inner product of length-$n_2$ vectors for each $\omega \in \mathbb{F}_q$.
- Complexity of branch metric calculation is $O(k_1 k_2 2^{k_2})$. 
APP Calculation (1/2)

- As usual we have

\[ \varphi_i(s) = \sum_{s'} \varphi_{i-1}(s') \gamma_i(s', s) \quad \text{and} \quad \beta_i(s) = \sum_{s'} \beta_{i+1}(s') \gamma_{i+1}(s, s') \]

- Let

\[ \varphi_i = (\varphi_i(0), \varphi_i(1), \ldots, \varphi_i(\alpha^{q-2})) \]
\[ \beta_i = (\beta_i(0), \beta_i(1), \ldots, \beta_i(\alpha^{q-2})) \]
\[ \gamma_i = (\gamma_i(0), \gamma_i(1), \ldots, \gamma_i(\alpha^{q-2})) \]
\[ L_i = (L_i(0), L_i(1), \ldots, L_i(\alpha^{q-2})) \]

then

\[ \varphi_i = \varphi_{i-1} \odot \gamma_i \]
\[ \beta_i = \beta_{i+1} \odot \gamma_{i+1} \]

where \( \odot \) denotes convolution.
Next

\[ L_i(\omega) = \sum_{s,s':s+s' = \omega} \varphi_{i-1}(s) \gamma_i(s, s') \beta_i(s') \]
\[ = \gamma_i(\omega) \sum_s \varphi_{i-1}(s) \beta_i(s + \omega) \]

so (\cdot denotes element-wise multiplication)

\[ L_i = \gamma_i \cdot (\varphi_{i-1} \otimes \beta_i) \]
\[ = \gamma_i \cdot \left( \left( \otimes_{j=1}^{i-1} \gamma_j \right) \otimes \left( \otimes_{j=i+1}^{n_1} \gamma_j \right) \right) \]
Using FFT

- To calculate $L_i$ we have to take the convolution of all vectors $\gamma_j$ but $\gamma_i$.
- In principle complexity of convolution scales as $O(q^2)$.
- However, complexity reduced to $O(q \log_2 q)$ by applying fast Fourier transform on finite Abelian groups (in this case equal to Hadamard transform):

$$L_i = \gamma_i \cdot H \left( \left( \cdot_{j=1}^{i-1} H(\gamma_j) \right) \cdot \left( \cdot_{j=i+1}^{m} H(\gamma_j) \right) \right)$$

- Complexity $O(k_1 2^{k_2})$ under Viterbi decoding is turned into $O(k_1 k_2 2^{k_2})$. 
Improving Multiplicity of Minimum Weight Codewords

- We have
  \[ d = 2d_2 \quad \text{and} \quad A_{\min} = \frac{n_1(n_1 - 1)}{2} A_{\min,2} \]

- To preserve \( d = 2d_2 \) while reducing \( A_{\min} \), we replace \( H = [1 \ 1 \ \cdots \ 1] \) with \( H' = [\beta_1 \ \beta_2 \ \cdots \ \beta_{n_1}] \) with \( \beta_i \in \mathbb{F}_q \setminus \{0\} \).

- Upon a uniformly random choice of \( \beta_1, \beta_2, \ldots, \beta_{n_1} \) we expect
  \[ \bar{A}'_{\min} = \frac{n_1(n_1 - 1)}{2} \frac{A_{\min,2}^2}{2^{k_2} - 1} \]

\((n_1, n_1 - 1)\) SPC over \( \mathbb{F}_q \)

\[ H = [\beta_1 \ \beta_2 \ \cdots \ \beta_{n_1}] \quad M : \mu_i \in \mathbb{F}_q \leftrightarrow x_i = 1 - 2c_i \]
Consider again the $(8, 7)$ SPC $\times (24, 12)$ Golay code.

Decoded by:

- The BCJR algorithm to the component code trellises, iterating the soft information exchange (50 iterations max);
- The BCJR algorithm, by weighting the soft-output of each component decoder by a factor $1/2$ [Pyndiah1998];
- The symbol-wise MAP decoder.

Additionally, we consider a second code (CC) with the same parameters but designed using the $H' = [\beta_1 \beta_2 \cdots \beta_{n1}]$ matrix approach.
Numerical Results

![Graph showing numerical results for different decoding methods. The x-axis represents \(E_b/N_0\) in dB, and the y-axis represents CER (Cumulative Error Rate). The graph compares:
- Product - Iterative
- Product - Iterative (scaling)
- Product - MAP
- Product - truncated UB
- CC - MAP
- CC - truncated UB

The graph indicates the performance of each method across various signal-to-noise ratios, showing how CER decreases as \(E_b/N_0\) increases, with the Product - MAP and Product - truncated UB methods generally showing the best performance.](image-url)
Conclusion

- Optimum decoding investigated for product codes given by concatenation of a binary SPC code with a low-dimension binary linear block code.
- Decoding complexity can be reduced further with respect to block-wise ML decoding by approaching the problem as a symbol-wise MAP decoding.
- A generalization of the code construction, enjoying the same low-complexity decoding principle is presented and analyzed, achieving additional coding gains.