9 Turboähnliche Kodes

9.1 LDPC[Low-Density Parity-Check]-(Block)Kodes

- 1962 von GALLAGER erstmals vorgestellt, trotz ihrer sehr guten Eigenschaften lange vergessen
  (Kodeverkettung schien überlegen, begrenzte Rechenleistung; FORNEY: „The way clearly far too complicated for the technology of the time.“ IEEE Spectrum, 2004.)
  1993 *Turbo Code*, zeigt Stärke iterativer Dekodierung auf
  1995 von MacKay/Neal leicht verbessert wiedererfunden und weiterentwickelt; (**L** zur Einordnung und Weiterentwicklung[10])
- Grundlage für iterative Dekodierungsalgorithmen:
  Sparsame Kontrollmatrix $H_{k\times n}$ mit $w(H) \ll k \cdot n$
  (Für einen „guten“ Kode sollte aber Generatormatrix nicht sparsam sein: $w(a^*) = 1 \rightarrow w(a) \gg 1$)
- Iterative Dekodierung kann Maximum-Likelihood(ML) Leistung erreichen (näherungsweise Umsetzung von ML)
- Unterscheidung von LDPC-Kodes
  - regulär: Spaltengewicht γ , Zeilengewicht ρ konstant, $\gamma < \rho$;
    $$R = 1 - \frac{\gamma}{\rho} = \frac{l}{n}; \quad \text{4-zyklenfrei: } d_{min} \geq \gamma + 1$$
  - irregulär: Spalten- γ_j , Zeilengewicht ρ_i unterschiedlich;
    $$R = 1 - \frac{\frac{1}{n}\sum_{j=1}^{n}\gamma_j}{\frac{1}{k}\sum_{i=1}^{k}\rho_i} = \frac{l}{n}$$

[10] E. Arikan, N. Hassan, M. Lentmaier et. al. *Challenges and some new directions in channel coding.* Journal of Comm. and Networks, August 2015

Beispiel

LDPC-Kode mit $\gamma = 3$, $\rho = 6$, $R = 1 - \frac{3}{6} = \frac{1}{2} = \frac{l}{n}$

$$H_{k\times n=6\times 12} = \begin{pmatrix} 1\,1\,0\,1\,0\,0\,0\,0\,1\,1\,0\,1 \\ 1\,1\,1\,1\,1\,0\,0\,1\,0\,0\,0\,0 \\ 1\,0\,1\,0\,1\,1\,0\,0\,0\,1\,1\,0 \\ 0\,0\,0\,1\,1\,1\,1\,0\,1\,0\,1\,0 \\ 0\,1\,1\,0\,0\,0\,1\,1\,0\,0\,1\,1 \\ 0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,0\,1 \end{pmatrix} = [H_l\, H_k]\,, \quad d_{min} = 4$$

In Vorbereitung der Anwendung iterativer Dekodierung:

- Zu H definierte Indexmengen:

  $K_i = \{j : 1 \le j \le n, h_{ij} = 1\} \quad (i = 1, 2, ..., k)$, $|K_i| = \rho_{[i]}$
  Menge aller Variablen (Bits), die Einfluss auf die i-te Prüfgleichung haben

  Beispiel: $K_1 = \{1, 2, 4, 9, 10, 12\}$, $K_2 = \{1, 2, 3, 4, 5, 8\}$, ...

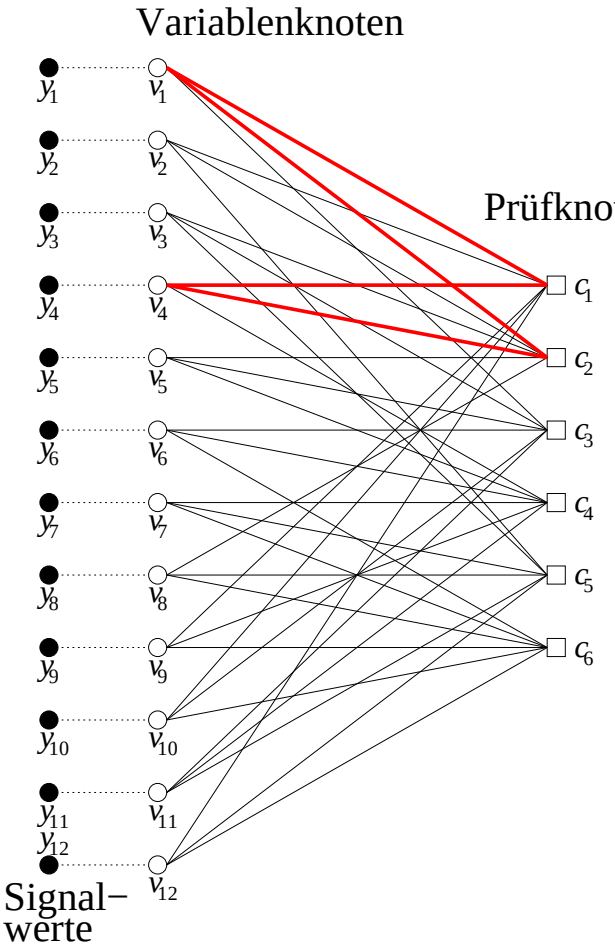
  $N_j = \{i : 1 \le i \le k, h_{ij} = 1\} \quad (j = 1, 2, ..., n)$, $|N_j| = \gamma_{[j]}$
  Menge der Prüfgleichungen, in welche die j-te Variable Einfluss hat
  Beispiel: $N_1 = \{1, 2, 3\}$, $N_2 = \{1, 2, 5\}$, $N_3 = \{2, 3, 5\}$, ...

  Anstelle $s = H \cdot b_h^T$ weniger Operationen mit

  $$s_i = \sum_{j\in K_i} y_{h,j} \bmod 2 \quad (i = 1, 2, ..., k),\ y_{h,j} \in \{0, 1\}$$

  (Einsparung von Multiplikationen, auch Additionen).

- Zu H korrespondierender TANNER-Graph [factor, bipartite graph]



Jede(s) Variable (Bit) v_j (u_j) ist in drei Prüfgleichungen enthalten, korrespondierend zu den drei Einsen in jeder Spalte.
Jede Prüfgleichung c_i basiert auf sechs Variablen, korrespondierend zu den sechs Einsen in jeder Zeile.

Optimal: Graph ohne Zyklen [girth] $\longrightarrow$ Leistung einer ML Dekodierung

Ist das Gewicht der stellenweisen Multiplikation von zwei beliebigen Spalten (Zeilen) höchstens Eins, dann ist H 4-zyklenfrei.

- H-Konstruktionen
  - zufallsähnlich (Vorgabe von Zyklenlänge, Gewichtsverteilung von Variablen-, Prüfknoten,..., z. B. mit PEG Verfahren)
  - strukturiert (algebraische Zshg.e, wie quasi-zyklische Kodes)

Kodierung (abhängig von H-Konstruktion)

- $a = a^* \cdot G = a_l \cdot G \quad (H \to G\,?)$
- $H_{k\times n} = [H_l\, H_k]$ und $a = [a_l\, a_k]$;
  Wegen $H \cdot a^T = \mathbf{0}$ ist $H_k \cdot a_k^T = H_l \cdot a_l^T$ und damit
  $\to a_k^T = \mathbf{H_k^{-1}} \cdot H_l \cdot a_l^T$
  (Voraussetzung: H_k quadratisch und invertierbar)
- H zyklisch:
  $a_j = \sum_{j' \in K_i \setminus j} a_{j'} \bmod 2 \quad (j = l + i,\ i = 1, 2, ..., k)$
- $H = [H_l\, H_k] = [H_1\, H_2]$
  H_1 zufallsähnlich; H_2 bidiagonal (staircase) (DVB-S2) oder
  H_2 untere Dreiecksmatrix (triangle):
  Kodierung wie H zyklisch

Dekodierung

Dekodierungsmethode:

Iterative (Syndrom-)Dekodierung $\quad s = H \cdot b_h^T = \mathbf{0}\,?$

Dekodierungsalgorithmen:

- hard-decision Dekodierung:
  Basis: $y_{h,j} \in \{0, 1\}$
- hard/soft reliability-based (hybride) Dekodierung:
  Basis: $y_{h,j} \in \{0, 1\}$ und bei soft-Zuverlässigkeit Einbeziehung der Signalwerte $y_j \in \mathbb{R}$ in den Berechnungsablauf
- soft-decision Dekodierung:
  Basis: $y_j \in \mathbb{R}$

hard-decision und soft/hard reliability-based Dekodierung

Beispiel: $a_l = (110100)$ $a_k = (010011)$ $\longrightarrow$

b_M =(-0.9,-0.9, 1.0, 0.1, 0.3, 1.0, 0.8,-0.7, 1.0, 0.9,-1.0, 0.4), $b_h^{(0)} \in A$?

- **bit-flipping** Algorithmus (GALLAGER 1962)

  $s_i^{(0)} = \sum_{j \in K_i} y_{h,j}^{(0)} \bmod 2 \quad (i = 1, 2, ..., k)$

  $s^{(\tau)} \neq \mathbf{0}: \; e_j^{(\tau)} = \sum_{i \in N_j} s_i^{(\tau)} \quad (j = 1, 2, ..., n)$

  $\text{flip}(y_{h,j}^{(\tau+1)})$, wenn $e_j^{(\tau)} \geq T$ oder $= T_{max} = \max_{j=1}^{n} e_j^{(\tau)}$

  Abbruch: $s^{(\tau+1)} = \mathbf{0}$ oder $\tau > \tau_{max}$, sonst $\tau = \tau + 1$

- hybride bit-flipping Algorithmen (2001 ... 2008 ...)

  – weigthed bit-flipping (WBF): bezieht Signalwerte y_j als soft-Zuverlässigkeit einer Prüfgleichung ein:

  $w_i = \min_{j \in K_i} |y_j| \quad (i = 1, 2, ..., k)$ (QWBF[11]: $w_i > \Delta : w_i = 2$, sonst $w_i = 1$)

  $s^{(\tau)} \neq \mathbf{0}: \; e_j^{(\tau)} = \sum_{i \in N_j} (2\, s_i^{(\tau)} - 1) \cdot w_i \quad (j = 1, 2, ..., n)$

  $\text{flip}(y_{h,j}^{(\tau+1)})$, wenn $e_j^{(\tau)} = T_{max}$

  – modified WBF (MWBF): bezieht Signalwerte als soft-Zuverlässigkeit einer Prüfgleichung und Bit-basiert ein (α – Wichtungsfaktor, exper. ermittelt, wächst mit γ: $\alpha \in [0.3, 2.0]$):

  $w_i = \min_{j \in K_i} |y_j| \quad (i = 1, 2, ..., k)$

[11] J. Zhang, et al. *Low-Latency Decoding of EG LDPC Codes.* Journal of Lightwave Technology, Vol. 25, No. 9, Sept. 2007

$$s^{(\tau)} \neq \mathbf{0}: \; e_j^{(\tau)} = \sum_{i \in N_j} (2\, s_i^{(\tau)} - 1) \cdot w_i - \alpha\, |y_j| \quad (j = 1, 2, ..., n)$$

$$\text{flip}(y_{h,j}^{(\tau+1)}), \text{ wenn } e_j^{(\tau)} = T_{max}$$

– improved MWBF (IMWBF): wie modified, bezieht aber extrinsische Information $L_e(\widehat{u}_{ij})$ als soft-Zuverlässigkeit ein:

  Die **extrinsische Information** $L_e(\widehat{u}_{ij})$ wird immer *unabhängig* vom Wert der Stelle j selbst berechnet und ist *abhängig von der Struktur des Kodes*, gegeben mit der Kontrollmatrix.

$$L_e(\widehat{u}_{ij}) = w_{ij} = \min_{j' \in K_i \setminus j} |y_{j'}| \quad (\forall j \in K_i,\, i = 1, 2, ..., k)$$

$$s^{(\tau)} \neq \mathbf{0}: e_j^{(\tau)} = \sum_{i \in N_j} (2 s_i^{(\tau)} - 1) w_{ij} - \alpha |y_j| \quad (j = 1, 2, ..., n)$$

$$\text{flip}(y_{h,j}^{(\tau+1)}), \text{ wenn } e_j^{(\tau)} = T_{max}$$

– iterative Anpassung der Prüfgleichungs-Zuverlässigkeit[12] über alle $j_{\text{flip}}^{(\tau)}$ nicht erfüllte Prüfgleichungen $s_i^{(\tau)} = 1$:

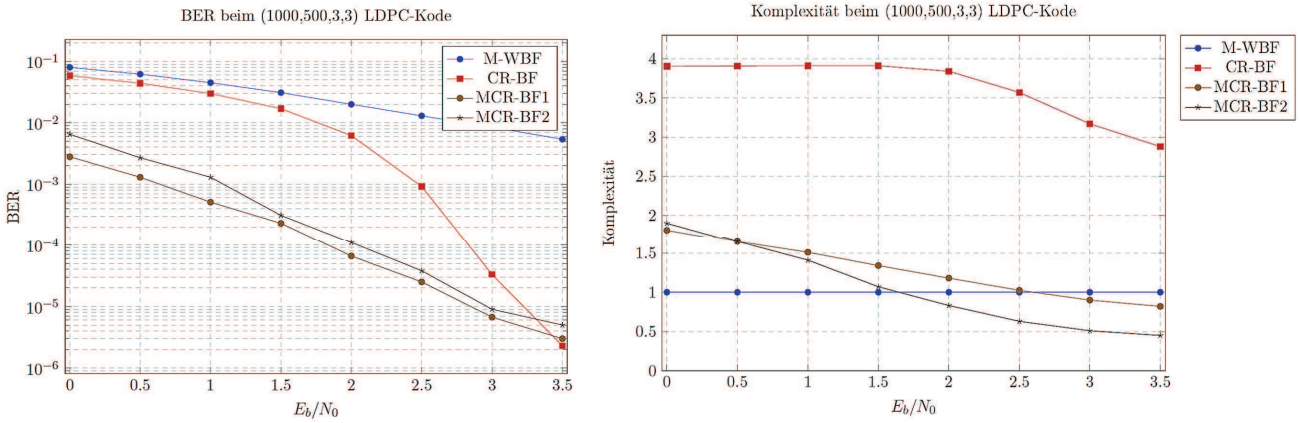
$$w_i^{(\tau+1)} = w_i^{(\tau)} + \delta \quad \text{bzw.} \quad w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} + \delta \qquad (\delta \approx 0.03)$$

⋮

[12] D. Qian, et al. *A Modification to Weighted Bit-Flipping Decoding Algorithm for LDPC Codes based on Reliability Adjustment.* 2008 IEEE

– C.Y. Chang, et. al. *Check Reliability Based Bit-Flipping Decoding Algorithms for LDPC Codes.* 2010, u.a. untersucht in:

A. Kropp *Einordnung und Bewertung jüngst modifizierter BF/MLG Dekodierungsalgorithmen.* Bachelorarbeit 2014



→ modifizierte Versionen (MCR-BF1,MCR-BF2) kippen in Abhängigkeit von $w(s)$ *mehrere Bits*!

MCR-BF1: $w(s) > 3\,\gamma$ → neben zu kippendem Bit werden 3 weitere Bits, die mit den nächst höchsten Kosten in $\mathbf{e}^{(\tau)}$, gekippt

MCR-BF2: kippen zusätzlicher $\lfloor \frac{w(s)}{2\,\gamma} \rfloor$ Bits

Kontrollmatrix **orthogonal**[13]: $\gamma = \rho$ in $\mathbf{H_{n \times n}}$, $d_{min} = \gamma + 1$
$\rightarrow$ **majority-logic** (MLG) Algorithmus

Beispiel: (15,7,5)BCH, orthogonal

$g(x) = x^8 + x^7 + x^6 + x^4 + 1 \rightarrow h(x) = \frac{f(x)}{g(x)} = x^7 + x^6 + x^4 + 1$
$a = (100010111000000) \quad \longrightarrow \quad b_h = (111010111000000) \in A?$

- one-step majority-logic Algorithmus (REED 1954, MASSEY 1963)
  Notw.: orthogonale Prüfsumme bezogen auf Variable (Bit) v_j

  $$s_i = \sum_{j \in K_i} y_{h,j} \bmod 2 \quad (i = 1, 2, ..., n)$$

  $$s \neq \mathbf{0}: \; e_j = \sum_{i \in N_j} s_i \rightarrow \text{flip}(y_{h,j}), \text{ wenn } e_j > \lfloor \tfrac{\gamma}{2} \rfloor \qquad (j = 1, 2, ..., n)$$

  Auch möglich:

  $$s \neq \mathbf{0}: \; e_j = \sum_{i \in N_j} (2\, s_i - 1) \text{ mit } e_j \in [-\gamma, +\gamma]$$
  $$\rightarrow \text{flip}(y_{h,j}), \text{ wenn } e_j > 0 \qquad (j = 1, 2, ..., n)$$

  $\rightarrow$ Überprüfung von $s = \mathbf{0}$? zum Erkennen von Dekodierungsversagen!

[13] Summiere v_j beeinflussende Prüfgleichungen: Summe über v_j ist γ, sonst höchstens 1.

- hard **r**eliability-based **iterativer** MLG Algorithmus (2009 [14])

  $r_j^{(0)} = \gamma, \;\; \text{wenn} \;\; y_{h,j}^{(0)} = 0, \;\; \text{sonst} \;\; r_j^{(0)} = -\gamma$

  $s_i^{(0)} = \sum_{j \in K_i} y_{h,j}^{(0)} \bmod 2 \quad (i = 1, 2, ..., n)$

  $s^{(\tau)} \neq \mathbf{0} : \left[L_e(\widehat{u}_{ij}) = \sum_{j' \in K_i \setminus j} y_{h,j'}^{(\tau)} \bmod 2 = s_i^{(\tau)} \oplus y_{h,j}^{(\tau)} \;\; (\forall j \in K_i, i = 1..n) \right]$

  $e_j^{(\tau)} = \sum_{i \in N_j} (2\, L_e(\widehat{u}_{ij}) - 1) = \sum_{i \in N_j} (2(s_i^{(\tau)} \oplus y_{h,j}^{(\tau)}) - 1)$

  $r_j^{(\tau+1)} = \min\{\max\{r_j^{(\tau)} - e_j^{(\tau)}, -\gamma\}, \gamma\}$

  $y_{h,j}^{(\tau+1)} = 0, \;\; \text{wenn} \;\; r_j^{(\tau+1)} \geq 0, \;\; \text{sonst} \;\; y_{h,j}^{(\tau+1)} = 1 \qquad (j = 1..n)$

  Abbruch: $s^{(\tau+1)} = \mathbf{0}$ oder $\tau > \tau_{max}$, sonst $\tau = \tau + 1$

- soft reliability-based iterativer MLG Algorithmus

  $x\,Bit$-Quantisierung: $|2^{x-1} - 1| \leq \gamma$!

  $r_j^{(0)} = y_{q,j} \in [-(2^{x-1}-1), 2^{x-1}-1], y_{q,j} = \text{round}\,(y_j \cdot (2^{x-1}-1)) \quad (j = 1..n)$

  min/max-Anpassung an Quantisierungsbereich, sonst wie hard

Vergleich WBF- und iterative MLG Algorithmen

– MLG erfordern nur logische Operationen und Integer-Additionen
– MLG erlauben parallele Dekodierung
– MLG nur für orthogonale H
– hard MLG vergleichbar mit BF, schlechter als WBF
– soft MLG konvergiert wesentlich schneller als WBF
– soft MLG nur < 1 dB zum optimalen sum-product Algorithmus

[14] Q. Huang, et al. *Two Reliability-Based Iterative Majority-Logic Decoding Algorithms for LDPC Codes.* 2009 IEEE

EXKURS: **Log-Likelihood-Algebra** [11,**20**,..]

Rechnen mit logarithmischen Wahrscheinlichkeiten

- **Zuverlässigkeit einer Bitentscheidung**

  $$L(u_j) = \ln \frac{P(u_j = 0)}{P(u_j = 1)} \;\rightarrow\; L(x_j) = \ln \frac{P(x_j = +1)}{P(x_j = -1)}$$

  $L(x)$ ist „soft"-Wert einer binären Zufallsvariablen.
  si gn $L(x)$ bestimmt die harte Entscheidung, $|L(x)|$ gibt die Zuverlässigkeit dieser Entscheidung an.

- **Zuverlässigkeit eines Empfangswertes**

  $$L(y_j|x_j) = \ln \frac{p(y_j|x_j = +1)}{p(y_j|x_j = -1)} \sim y_j = \operatorname{sign} y_j \cdot |y_j|$$

  Je größer $|y_j|$ umso zuverlässiger die harte Entscheidung sign y_j.
  Gelingt es $|y_j|$ in den Dekodierungsalgorithmus einzubringen, spricht man von soft-decision Dekodierung.[15]

- **Regel von BAYES**

  $$L(\widehat{x}_j) = L(x_j|y_j) = \ln \frac{P(x_j = +1|y_j)}{P(x_j = -1|y_j)}$$
  $$= \ln \frac{p(y_j|x_j = +1)}{p(y_j|x_j = -1)} + \ln \frac{P(x_j = +1)}{P(x_j = -1)} = L_c\, y_j + L(x_j)$$

  L_c Kanalzuverlässigkeit; SBK: $L_c = \ln \frac{1-p_s}{p_s}$, AWGN: $L_c = \frac{2}{\sigma^2}$
  $L(y|x) = L_c\, y = L(x|y) - L(x)$
  Zuverlässigkeit des Empfangswertes ist die Differenz zwischen a posteriori und a priori soft-Wert von x (nach – vor der Ü).

[15] $y_{h,j} = \frac{1-\operatorname{sign} y_j}{2}$

- **Zuverlässigkeit der Summe**[16] zweier statistisch unabhängiger Werte x_1 und x_2: (paarweise Auswertung)

$$L(x_1 \oplus x_2) = \ln \frac{1 + e^{L(x_1)} e^{L(x_2)}}{e^{L(x_1)} + e^{L(x_2)}}$$

$$= \operatorname{sign} L(x_1) \cdot \operatorname{sign} L(x_2) \cdot \min\{|L(x_1)|, |L(x_2)|\}$$
$$+ \ln(1 + e^{-|L(x_1)+L(x_2)|}) - \ln(1 + e^{-|L(x_1)-L(x_2)|})$$

Approximationen für $\ln(1 + e^{-|z|})$:

$\rightarrow$ 0 ; $L(x_1 \oplus x_2) \approx \operatorname{sign} L(x_1) \cdot \operatorname{sign} L(x_2) \cdot \min\{|L(x_1)|, |L(x_2)|\}$

$\rightarrow$ Lookup-Tabelle:

$0.000 \leq \lvert z\rvert < 0.196$	0.65
$0.196 \leq \lvert z\rvert < 0.433$	0.55
$0.433 \leq \lvert z\rvert < 0.710$	0.45
$0.710 \leq \lvert z\rvert < 1.050$	0.35
$1.050 \leq \lvert z\rvert < 1.508$	0.25
$1.508 \leq \lvert z\rvert < 2.252$	0.15
$2.252 \leq \lvert z\rvert < 4.500$	0.05
$4.500 \leq \lvert z\rvert < +\infty$	0.00

$\rightarrow$ Lineare Abbildung:

$0.0 \leq \lvert z\rvert < 0.5$	$-\lvert z\rvert \cdot 2^{-1} + 0.7000$
$0.5 \leq \lvert z\rvert < 1.6$	$-\lvert z\rvert \cdot 2^{-2} + 0.5750$
$1.6 \leq \lvert z\rvert < 2.2$	$-\lvert z\rvert \cdot 2^{-3} + 0.3750$
$2.2 \leq \lvert z\rvert < 3.2$	$-\lvert z\rvert \cdot 2^{-4} + 0.2375$
$3.2 \leq \lvert z\rvert < 4.4$	$-\lvert z\rvert \cdot 2^{-5} + 0.1375$
$4.4 \leq \lvert z\rvert < +\infty$	0

Bezogen auf die Signalwerte y_j:

$\longrightarrow L(y_1 \oplus y_2 \oplus ...) \approx \prod_j \operatorname{sign} y_j \cdot \min_j |y_j|$

---

[16] $L(x_1 \oplus x_2) = \ln \frac{P(x_1 \oplus x_2 = 0)}{P(x_1 \oplus x_2 = 1)}$

$$= \ln \frac{P(x_1 = 0)P(x_2 = 0) + P(x_1 = 1)P(x_2 = 1)}{P(x_1 = 0)P(x_2 = 1) + P(x_1 = 1)P(x_2 = 0)}$$

$$= \ln \frac{e^{L(x_1)} P(x_1 = 1)\, e^{L(x_2)} P(x_2 = 1) + P(x_1 = 1)P(x_2 = 1)}{e^{L(x_1)} P(x_1 = 1)\, P(x_2 = 1) + P(x_1 = 1)\, e^{L(x_2)} P(x_2 = 1)}$$

$$= \ln \frac{e^{L(x_1)} e^{L(x_2)} + 1}{e^{L(x_1)} + e^{L(x_2)}}$$

NR: $L(x) = \ln \frac{P(x=0)}{P(x=1)} \rightarrow P(x = 0) = e^{L(x)} P(x = 1)$

soft-decision Dekodierung

- **optimal**er sum-product Algorithmus,
  auch message-passing, belief-propagation (BP)

  Basis: Wahrscheinlichkeiten oder Zuverlässigkeiten;

  Zuverlässigkeit der Summe zweier Signalwerte:

$$L(y_1 \oplus y_2) = \ln \frac{1 + e^{y_1} e^{y_2}}{e^{y_1} + e^{y_2}}$$

$$= \operatorname{sign} y_1 \cdot \operatorname{sign} y_2 \cdot \min\{|y_1|, |y_2|\}$$

$$\underbrace{+ \ln(1 + e^{-|y_1+y_2|}) - \ln(1 + e^{-|y_1-y_2|})}_{\text{Lookup-Tabelle, stückweise lineare Abbildung, ...}}$$

- **sub-optimal**er min-sum Algorithmus

$$L(y_1 \oplus y_2) \approx \operatorname{sign} y_1 \cdot \operatorname{sign} y_2 \cdot \min\{|y_1|, |y_2|\}$$

$$\longrightarrow L(y_1 \oplus y_2 \oplus ...) \approx \prod_j \operatorname{sign} y_j \cdot \min_j |y_j|$$

- **quasi-optimal**e min-sum Algorithmen zur Wiederherstellung des Leistungsverlustes von min-sum zu sum-product:

  normalized: $\alpha = 1.25...2.00$ (abh. von H-Konstruktion und γ)
  $L(y_1 \oplus y_2) \approx \frac{1}{\alpha} \cdot \operatorname{sign} y_1 \cdot \operatorname{sign} y_2 \cdot \min\{|y_1|, |y_2|\}$

  offset: $\beta = 0.15...0.20$
  $L(y_1 \oplus y_2) \approx \operatorname{sign} y_1 \cdot \operatorname{sign} y_2 \cdot \max\{\min\{|y_1|, |y_2|\} - \beta, 0\}$

  self-corrected (2008):
  bei VZ Wechsel zwischen zwei Iterationen Nullsetzung (Auslöschung) → Experimentierbaustein siehe web Kanalkodierung

  ... weitere Modifizierungen

Zur Einordnung der Algorithmen:

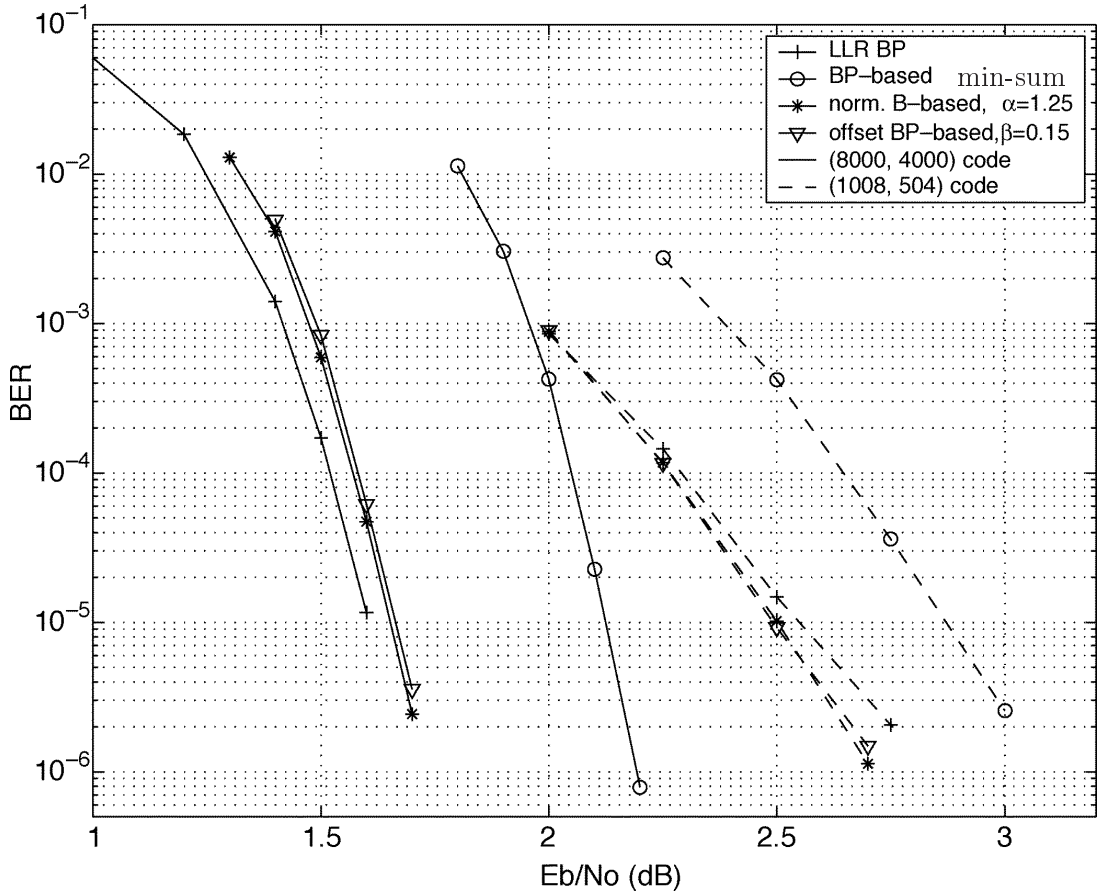


Fig. 2. Bit-error performances of an (8000,4000) and a (1008,504) LDPC code under two improved BP-based algorithms ($\mathrm{it}_{\max} = 100$).

J. Chen, A. Dholakia, E. Eleftheriou, M. P.C. Fossorier. *Reduced-Complexity Decoding of LDPC Codes.* IEEE Trans. on Comm., Vol. 53, No. 8, Aug. 2005, S. 1293

$\longrightarrow$ bei kurzen/mittleren Kodewortlängen ist sum-product wegen vorhandener Korrelationen zwischen den Nachrichten nicht optimal

$\longrightarrow$ bei großen Kodewortlängen liegt Abstand zwischen quasi-optimalen und optimalem Algorithmus bei 0,5 ... 0,6 dB

soft-decision Umsetzungen

• optimaler **sum-product** Algorithmus

$$a \in \{0,1\}^n \rightarrow a_M \in \{+1,-1\}^n \rightarrow b_M \in \mathbb{R}^n \rightarrow b \in \mathbb{R}[0,1]^n$$
$$u_j \in \{0,1\} \rightarrow x_j \in \{+1,-1\} \rightarrow y_j \in \mathbb{R} \rightarrow p(y_j) \in \mathbb{R}[0,1]$$

$$y_j = x_j + z_j \quad \text{mit} \quad z_j = ZZ_{NV[0,\sigma^2]}\,, \; \sigma = \frac{1}{\sqrt{2\,R\,10^{\frac{E_b}{N_0}[dB]/10}}}$$

(AWGN-Modell mit Kanalzuverlässigkeit $L_c = \frac{2}{\sigma^2}$;
Algorithmus erfordert Kenntnis von L_c !)

und damit

$$p(y_j) = \frac{1}{1+e^{-\frac{2}{\sigma^2}\,y_j}} \quad (j = 1,2,...,n)$$

und

$$y_{h,j} = \begin{cases} 0 & p(y_j) \geq 0.5 \\ 1 & \text{sonst} \end{cases}$$

$\longrightarrow$ **Algorithmus** auf Basis von Wahrscheinlichkeiten [probability domain]

Aufbau von 6 Matrizen:
$Q^0_{k\times n}, Q^1_{k\times n}, \Delta Q_{k\times n}, \Delta R_{k\times n}, R^0_{k\times n}, R^1_{k\times n}$

$\longrightarrow$ Algorithmus in Anlehnung an MacKay/Neal [27], 1996

$\forall i, j \wedge h_{ij} = 1$:

	1. Iteration	>1. Iteration
vertikal	$q_{ij}^0 = p(y_j)$ $q_{ij}^1 = 1 - q_{ij}^0$	$q_{ij}^0 = p(y_j) \prod_{i' \in N_j \setminus i} r_{i'j}^0 = q_j^0 / r_{ij}^0$ $q_{ij}^1 = (1 - p(y_j)) \prod_{i' \in N_j \setminus i} r_{i'j}^1 = q_j^1 / r_{ij}^1$ Skaliere: $q_{ij}^0 + q_{ij}^1 = 1$
horizontal	$\delta q_{ij} = q_{ij}^0 - q_{ij}^1$ $\delta r_{ij} = \prod_{j' \in K_i \setminus j} \delta q_{ij'}$ $r_{ij}^0 = \frac{1}{2}(1 + \delta r_{ij})$ und $r_{ij}^1 = \frac{1}{2}(1 - \delta r_{ij}) = 1 - r_{ij}^0$	
Auswertung	$q_j^0 = p(y_j) \prod_{i \in N_j} r_{ij}^0$ und $q_j^1 = (1 - p(y_j)) \prod_{i \in N_j} r_{ij}^1$ Skaliere, so dass $q_j^0 + q_j^1 = 1$ mit $q_j^0 = q_j^0 \cdot \frac{1}{q_j^0 + q_j^1}$ und $q_j^1 = 1 - q_j^0$ $s = H \cdot b_{korr,h}^T = H \cdot (q_{h,1}^0, q_{h,2}^0, ..., q_{h,n}^0)^T = \mathbf{0}$? ja: stop bzw. τ_{max} erreicht nein: weiter mit vertikalen Schritt	

Beispiel $p(y_j|u=0) \in \mathbb{R}[0,1]$:

u_{11} 1	u_{12} 0	p_1^- 1
u_{21} 0	u_{22} 1	p_2^- 1
$p_1^\|$ 1	$p_2^\|$ 1	

→

y_1 0.6	y_2 0.6	y_3 0.1
y_4 0.8	y_5 0.7	y_6 0
y_7 0.4	y_8 0	

$\longrightarrow H_{k \times n}$?

Weiter: $y_j \sim \ln \frac{p(y_j|u=0)}{p(y_j|u=1)} = \ln \frac{p(y_j|u=0)}{1 - p(y_j|u=0)}$

• sub-optimaler **min-sum** Algorithmus

$a \in \{0,1\}^n \rightarrow a_M \in \{+1,-1\}^n \rightarrow b_M \in \mathbb{R}^n \rightarrow b \in \mathbb{R}^n$

$\longrightarrow$ **Algorithmus** auf Basis von Zuverlässigkeiten [log domain]

$$L(y_1 \oplus y_2 \oplus ...) \approx \prod_j \operatorname{sign} y_j \cdot \min_j |y_j|$$

Aufbau von 2 Matrizen: $Q_{k\times n}, R_{k\times n}$

$\forall i,j \wedge h_{ij} = 1$:

vertikal	1. Iteration	> 1. Iteration
	$L(q_{ij}) = y_j$	$L(q_{ij}) = y_j + \sum_{i' \in N_j \setminus i} L_e(r_{i'j}) = L(\widehat{u}_j) - L_e(r_{ij})$
horizontal	$L_e(r_{ij}) = \prod_{j' \in K_i \setminus j} \operatorname{sign} L(q_{ij'}) \cdot \min_{j' \in K_i \setminus j} \lvert L(q_{ij'})\rvert$	
Auswertung	$L(\widehat{u}_j) = y_j + \sum_{i \in N_j} L_e(r_{ij})$ $\widehat{u}_j = \begin{cases} 0 & L(\widehat{u}_j) \geq 0 \\ 1 & \text{sonst} \end{cases} \quad (j = 1,2,...,n)$ $s = H \cdot b_{korr,h}^T = H \cdot (\widehat{u}_1, \widehat{u}_2, ..., \widehat{u}_n)^T = \mathbf{0}$? ja: stop bzw. τ_{max} erreicht nein: weiter mit vertikalem Schritt	

Beispiel (Fortsetzung von Beispiel S. 152), $y_j \in \mathbb{R}$

$b = (+0.4 \;\; +0.4 \;\; -2.2 \;\; +1.4 \;\; +0.8 \;\; -7.0 \;\; -0.4 \;\; -7.0)$

Auch möglich: Auslöschungskorrektur!

$b_{ausl} = (-1 \;\; +1 \; \mathbf{0} \; \mathbf{0} \;\; -1 \;\; -1 \; \mathbf{0} \;\; -1)$; $y_j \in \{ \underbrace{+1,-1}_{u_j = \frac{1-\operatorname{sign} y_j}{2}}, \mathbf{0} \}$

- quasi-optimale **min-sum** Algorithmen (s.a. S. 149)
  $a \in \{0,1\}^n \rightarrow a_M \in \{+1,-1\}^n \rightarrow b_M \in \mathbb{R}^n \rightarrow b \in \mathbb{R}^n$

Im Vergleich zum min-sum Algorithmus NUR Modifizierung des **horizontal**en Schritts!

normalized: $\alpha = 1.25...2.00$ (abh. von H-Konstruktion und γ)

$$L_e(r_{ij}) = \frac{1}{\alpha} \prod_{j' \in K_i \backslash j} \operatorname{sign} L(q_{ij'}) \cdot \min_{j' \in K_i \backslash j} |L(q_{ij'})|$$

offset: $\beta = 0.15...0.20$

$$L_e(r_{ij}) = \prod_{j' \in K_i \backslash j} \operatorname{sign} L(q_{ij'}) \cdot \max\{\min_{j' \in K_i \backslash j} |L(q_{ij'})| - \beta, 0\}$$

Anwendungen

- DVB-S2,-T2,-C2 (Verkettung von BCH und LDPC, BCH entfernt möglichen error-floor), 0,7 ... 1,2 dB von Grenze, Koderate-Änderungen durch Punktierung von Paritätsstellen
- DÜ-Standards WLAN ($\leq 50\,m$) und WiMAX ($\leq 50\,km$)
- Verteilte Speichersysteme wie Cloud Computing, Behandlung von Auslöschungsbereichen (hybrid: Iterative Dekodierung/Gauß-Jordan-Elimination)
- IEEE 802.3an Standard für 10-Gigabit-Ethernet (Verkettung von CRC und (2048,1723)LDPC)
- Optische Netzwerke mit bit-flipping Algorithmus

$\longrightarrow$ SHANNON-Grenze nahezu erreicht ($< 1\,dB$), es geht „nur“ noch um praktikable Anwendungen

10 Schlussbetrachtungen

Einordnung und Besonderheiten betrachteter Kodes

- Ende der **40er Jahre** bahnbrechende Arbeiten von **SHANNON, HAMMING** und **GOLAY**
  - Annahmen Shannon's: Zufallskodes großer Länge, ML Dekodierung
    Problem: Dekodierungskomplexität wächst exponentiell mit n
  - Shannon hat nur theoretische Abhandlungen geführt, Möglichkeiten der Umsetzung offen gelassen
- Traditionelle, klassische algebraische Kodes der **50er, 60er Jahre**
  - Z. B. HAMMING, GOLAY, RM, BCH, RS, ... (hard-decision Dekodierungsalgorithmen)
    Faltungskodes (hard/soft-decision Dekodierungsalgorithmen)
  - Ein wesentlicher Vorteil von Faltungskodes gegenüber Blockkodes:
    Alle Synchronisationsprobleme sind mit VITERBI harmlos und mit geringem Mehraufwand zu beherrschen. UND: Die Dekodierung beginnt mit dem Einlesen der ersten Empfangssequenz. (Bei Blockkodes muss erst die Empfangsfolge komplett empfangen sein; Problem Strukturverzögerung.)
  - Komplexität der Dekodierung wächst mit n und d_{min}/d_f
  - Weit entfernt von SHANNON-Grenze $\longrightarrow$ **Kodeverkettung**, Verbesserung, aber nicht erkennbaren Sprung Richtung Grenze, erreichbar?
  - Anwendungen im Bereich kleiner und mittlerer Kodewortlänge
    (Audio-CD: $n = 256\,Bit$, GSM: $n = 456\,Bit$, DVB: $n = 1632\,Bit$, NASA: $n = 2040\,Bit$, UMTS: $n \approx 5000\,Bit$)
  - Traditionelle, klassische Verkettungen wie RS–Faltung galten **bis Mitte der 90er Jahre** als perfekt passend für die Fehlerkorrektur:

RS-Kode	Faltungskode
hohe Koderate	kleine Koderate
$R = 0.87$ (NASA)	$R = \frac{1}{2}$, $K = 7$
$R = 0.87$ (Mars)	$R = \frac{1}{6}$, $K = 15$
$p_R \approx 10^{-10}...10^{-12}$	für's „grobe“: $p_R < 10^{-5}$

Bei Fehlern über f_k hinaus produziert der Viterbi-Dekodierer Bündelfehler, während der BMD-Dekodierer von BCH-/RS-Kodes in den meisten Fällen Dekodierungsversagen aufzeigt (für $k_1 \geq 4, f_k \geq 4$ bis zu 100%).

Faltungskodes sind Blockkodes überlegen, wenn ein Kanal mit schlechter bis mittelmäßiger Qualität um 2 oder 3 Zehnerpotenzen in der Fehlerrate verbessert werden soll. Wenn dagegen extrem kleine Fehlerraten angestrebt werden, sind BCH-/RS-Kodes vorzuziehen.

$\longrightarrow$ Ausnutzen in der Kodeverkettung!

- Seit Mitte der **90er Jahre** Lücke zwischen Leistung praktikabler Kodes und SHANNON-Grenze mit Anwendung **Iterativer Dekodierung** (GALLAGER **1962!**) schließbar;

  Turbo Code (1993) hat Gebiet der Fehlerkorrekturkodes revolutioniert!

  - Grundlage: Umsetzung von Shannon's Annahmen
  - Turbokode: Verkettung algebraischer Kodes, Zufallsinterleaving, große Kodewortlänge $\longrightarrow$ Zufallskode
  - LDPC-Kode: Sparsame „zyklenfreie“ Kontrollmatrix

    (Zufallsmatrix großer Länge – Annahme, dass diese frei von kurzen Zyklen –, quasi-zyklische Matrix, zyklische Matrix mit orthogonaler Eigenschaft, PEG (Progressive Edge-Growth) Matrix, ...)
  - Iterative Dekodierung umgeht exponentielle Komplexität der ML Dekodierung!

    Ist (Näherungs-)Lösung für ML Umsetzung bei Blockkodes!
  - (Noch ?) Problem: Komplexität von Kodierung und Dekodierung

Komplexität	Turbokode	LDPC-Kode
Kodierung	linear	quadratisch/linear (Basis: G/H)
Dekodierung	quadratisch	linear
Koderate	$< \frac{1}{2}$	$\geq \frac{1}{2}$

Turbokodes mit eher kurzem Interleaver sind eine Option für zuverlässige Übertragung, wie am Beispiel UMTS die Datenübertragung. Kurz bedeutet hier eine kurze Dekodierungsverzögerung.[17]

- Finden praktikabler Umsetzungen nahe SHANNON-Grenze;
  „alte“ Dekodierungsverfahren wegen geringer Komplexität wieder interessant: bit-flipping (1962), majority-logic (1954/1963)
  $\longrightarrow$ u. a. soft reliability-based iterativer MLG Algorithmus (2009) ermöglicht parallele Dekodierung, erfordert nur logische Operationen und Integer-Additionen, konvergiert schnell (nur wenige Iterationen), nur 0,5 ... 1,0 dB von Leistung des sum-product Algorithmus; Anwendung?
- Große Kodewortlängen, z. B. DVB-S2: $n = 64800\,Bit$ oder bei zeitkritischeren Anwendungen $n = 16200\,Bit$
  ABER auch WiMAX (IEEE 802.16e): $n = 576 \ldots 2304\,Bit$, WLAN (IEEE 802.11n): $n = 648,\ 1296,\ 1944\,Bit$
  $\longrightarrow$ Grundlage bilden suboptimale quasi-zyklische H-Konstruktionen
- Rekonstruktion über f_k hinaus, keine Garantie von $\leq f_k$ (error-floor bei $p_R \approx 10^{-5} \ldots 10^{-6}$)
  $\longrightarrow$ Verkettung mit klassischen Kodes wie BCH, RS ($p_R \leq 10^{-10}$)
  (DVB-S2: Verkettung von BCH und LDPC,
  UMTS, LTE: Verkettung von RS und Turbokode,
  ...)

LDPC-Kodes ersetzen weder Turbokodes noch traditionelle, klassische Block- und Faltungskodes[18] (letztere haben begrenzte Dekodierungskomplexität, $p_R \ll$, bei hohen Datenraten unschlagbar)!

[17] T. Hehn, J.B. Huber. *LDPC Codes and Convolutional Codes with Equal Structural Delay: A Comparison.* IEEE Trans. on Comm., Juni 2009

[18] K.S. Andrews, D. Divsalar, ... *The Development of Turbo and LDPC Codes for Deep-Space Applications.* Proc. of the IEEE, Vol. 95, No. 11, Nov. 2007.