Iterative, sub-optimale Dekodierung von BCH-Kodes.

Prüfungsvorleistung in der Lehrveranstaltung Kanalkodierung

Hannes Weisbach

hannes.weisbach@tu-dresden.de

28. Oktober 2014

1 Aufgabenstellung

Wenden Sie auf BCH-Beschreibungen ($k_1 = 5$ oder ...; $g(x) \rightarrow h(x) \rightarrow H$) ein iteratives Dekodierungsverfahren an (sub-optimal oder quasi-optimal)! Ist mit diesen Realisierungen eine Rekonstruktion über f_k hinaus möglich? Werten Sie das Leistungsverhalten aus!

Verwenden Sie AWGN Rauschen:

$y_j = x_j + z_j$ mit $x_j \in +1, -1$, $z_j = Z$, $y_j \in \mathbb{R}$.

Die Zufallsvariable Z ist (μ, σ^2)-normalverteilt mit dem Erwartungswert $\mu = 0$ und der Rauschvarianz σ^2. Die Rauschvarianz σ^2 für $\frac{E_b}{N_0}$ in [dB] ergibt sich mit $\sigma^2 = \frac{1}{2R10^{\frac{E_b}{N_0}[\text{dB}]/10}}$.

Denkbares Szenario:

$a = \mathbf{0} \rightarrow a_M = \{+1\}^n$, N Simulationen für b_M je $\frac{E_b}{N_0}$[dB]:

$b = b_M : \frac{|b_{korr} \neq a|}{N}$
$b = b_h : \frac{|d(\mathbf{a}, \mathbf{b}_h) = w(\mathbf{e}) > f_k|}{N}$

Hinweis: $|x|$ – Anzahl der Elemente der entsprechenden Bedingung (Kardinalität).

Grafik WER $= f(\frac{E_b}{N_0}[\text{dB}])$? (WER[word error rate])

2 Lösung

2.1 BCH-Kodes

Mit den Parameter $k_1 = 5$ und dem irreduziblen, primitiven Modularpolynom $M(x) = x^5 + x^2 + 1$ ergibt sich eine Kodewortlänge von $n = 2^{k_1} - 1 = 31$. Mit der weiteren Einschränkung von $\mu \in \{0, 1\}$ und dem Abstand eins der Exponenten α^i der Nullstellen im Generatorpolynom $g(x)$ finden sich folgende BCH-Kodes:

μ	Kode	$g(x)$
0	(31, 30, 2)	$m_0(x)$
1	(31, 26, 3)	$m_1(x)$
0	(31, 25, 4)	$m_0(x) * m_1(x)$
1	(31, 21, 5)	$m_1(x) * m_3(x)$
0	(31, 20, 6)	$m_0(x) * m_1(x) * m_3(x)$
1	(31, 16, 7)	$m_1(x) * m_3(x) * m_5(x)$
0	(31, 15, 8)	$m_0(x) * m_1(x) * m_3(x) * m_5(x)$
1	(31, 11, 11)	$m_1(x) * m_3(x) * m_5(x) * m_7(x)$
0	(31, 10, 12)	$m_0(x) * m_1(x) * m_3(x) * m_5(x) * m_7(x)$
1	(31, 6, 15)	$m_1(x) * m_3(x) * m_5(x) * m_7(x) * m_{11}(x)$
0	(31, 5, 16)	$m_0(x) * m_1(x) * m_3(x) * m_5(x) * m_7(x) * m_{11}(x)$
1	(31, 1, 31)	$m_1(x) * m_3(x) * m_5(x) * m_7(x) * m_{11}(x) * m_{15}(x)$

Für die weiteren Betrachtungen wähle ich den (31, 16, 7)-BCH-Kode aus.

2.2 Sub-optimale iterative Dekodierung

Iterative Dekodieralgorithmen sind vor allem von LDPC-Kodes [3] her bekannt. Diese Algorithmen können jedoch auch auf allgemeine Kontrollmatrizen von beliebigen Blockkodes angewandt werden, die nicht unbedingt dünn besetzt sind oder den Bildungsvorschriften von LDPC-Kodes entsprechen. Dementsprechend sind Komplexitäts- oder Restfehlerschranken [3] nicht mehr unbedingt gültig. Iterative Algorithmen lassen sich grob in drei Kategorien einteilen: hard- und soft-decision sowie hybride Dekodierer [6].

Hard-decision-Dekodierer quantisieren das Eingangssignal in genau zwei Zustände. Dadurch geht Information verloren, was sich in schlechteren Dekodiereigenschaften auswirkt. Die Implementierung ist allerdings sehr einfach.

Soft-decision-Dekodierer quantisieren das Eingangssignal gar nicht, wodurch die Dekodiererkomplexität sehr hoch wird. Der Dekodierer kann allerdings bis Nahe an die SHANNON-Grenze dekodieren. Nur soft-decision-Dekodierer können überhaupt optimal sein.

Hybride Dekodierer verarbeiten nicht nur die quantisierte Information sondern auch eine Zuverlässigkeitsinformation, die aus der Empfangsfolge extrahiert wird. Ziel hybrider Dekodierer ist eine Dekodiererleistung nahe der SHANNON-Grenze bei möglichst geringer Komplexität. Hier sind suboptimale und quasi-optimale Algorithmen zu finden.

Maximum-Likelihood Dekodierung ist optimal für LDPC-Kodes, allerdings im Allgemeinen zu komplex. Daher werden quasi-optimale Algorithmen wie *belief propagation* (auch bekannt als *sum-product*-Algorithmus) benutzt. Durch approximieren der Summe durch das Maximum entsteht der *max-product*-Algorithmus. Da diese ebenfalls noch sehr rechenintensiv sind, werden die Berechnungen in der logarithmischen Domäne ausgeführt. In der logarithmischen Domäne entspricht ein Produkt einer Addition. Konsequenterweise wird dieser Algorithmus *max-sum*-Algorithmus genannt. Verwendet man jetzt negative Log-Likelihoods, wird max durch min ersetzt und man erhält den *min-sum*-Algorithmus.

Diese Algorithmen sind in etwa mit dem Viterbi-Algorithmus äquivalent. Das finden des Minimums von Summen entspricht dem Aktualisieren der Pfadmetrik und Auswahl der besten Pfadmetrik in jedem Schritt des Viterbi-Algorithmus.

Durch die verwendete Approximation haben die Algorithmen eine stark reduzierte Komplexität, aber auch ein reduziertes Korrekturvermögen im Vergleich zu optimalen oder quasi-optimalen Algorithmen.

Weitere Möglichkeiten sind *bit flipping* und *majority logic* Algorithmen von denen *hard-decision* und *soft-decision* Varianten existieren.

Ich habe mich für den *min-sum*-Algorithmus entschieden. Als Vergleich dient eine hard-decision-Dekodierung mittels Peterson-Gorenstein-Zierler-Algorithmus (PGZ) [4]. Zudem werte ich die Modifikationen *self-correcting Min-Sum* (SCMS) [10, 8] (zwei Variationen), *normalized Min-Sum* (NMS) [1], *offset Min-Sum* (OMS) [1] und *2D-normalized Min-Sum* [9] für BCH-Kodes aus.

Normalized, *offset* und *2D-normalized min-sum*-Algorithmen ist gemein, dass sie Parameter einführen, die von der Kanalcharakteristik und/oder dem Kode abhängen. Üblicherweise werden diese Parameter analytisch mittels *density evolution* für einen gegebenen Kode und Kanalcharakteristik optimiert. Es existieren auch iterative BP Dekodierer, die keine Information über die Kanalcharakteristik benötigen. Neben den untersuchten *self-correcting* Varianten sei hier der *universal most powerful* (UMP) Algorithmus [2] genannt, welcher unabhängig von N_0 ist.

Darüber hinaus gibt es noch adaptive Algorithmen [9, 5, 7], welche Parameter in Abhängigkeit vom aktuellen Ergebnis einer Paritätsgleichung oder SNR am Empfänger während der Laufzeit anpassen.

2.3 Kontrollmatrix für BCH-Kodes

Für den gewählten Kode lautet das Hauptpolynom $f(x) = x^n + 1 = x^{31} + 1$ und das Generatorpolynom

$$g(x) = m_1(x) * m_3(x) * m_5(x) = x^{15} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^5 + x^3 + x^2 + x + 1.$$

Das Kontrollpolynom lautet demnach

$$h(x) = \frac{f(x)}{g(x)} = x^{16} + x^{12} + x^{11} + x^{10} + x^9 + x^4 + x + 1.$$

Die Kontrollmatrix $H_{k\times n}$ ergibt sich aus der k-fachen zyklischen Verschiebung der n-stelligen binären Koeffizientendarstellung von $h(x)$:

$$H_{k\times n} = \begin{pmatrix} h(x) & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & h(x) \end{pmatrix}$$

Für den gewählten (31, 16, 7)-BCH-Kode ergibt sich folgende Kontrollmatrix:

$$H_{16\times 31} = \begin{pmatrix}
1&0&0&0&1&1&1&1&0&0&0&0&1&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&1&1&1&1&0&0&0&0&1&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&1&1&1&1&0&0&0&0&1&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&1&1&1&1&0&0&0&0&1&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&1&1&1&1&0&0&0&0&1&0&0&1&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&1&1&1&1&0&0&0&0&1&0&0&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&1&1&1&1&0&0&0&0&1&0&0&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&1&1&1&1&0&0&0&0&1&0&0&1&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&1&1&1&1&0&0&0&0&1&0&0&1&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&1&1&1&1&0&0&0&0&1&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&1&1&1&0&0&0&0&1&0&0&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&1&1&1&0&0&0&0&1&0&0&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&1&1&1&0&0&0&0&1&0&0&1&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&1&1&1&0&0&0&0&1&0&0&1&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&1&1&1&0&0&0&0&1&0&0&1&1
\end{pmatrix}.$$

2.4 Parameterbestimmung

Für *normalized*, *offset* und *2D-normalized Min-Sum* hängen die jeweiligen Parameter α und β vom verwendeten Kode und Kanal ab. Wie schon erwähnt werden diese Parameter normalerweise mittels *density evolution* bestimmt.

Dazu werden die Wahrscheinlichkeitsdichten für $p(L = l|x = 0)$ berechnet. Für jede Iteration des Dekodieralgorithmus kann aus der Wahrscheinlichkeitsdichte die Fehlerwahrscheinlichkeit errechnet werden. Mit einem Kanalmodell kann eine iterative Vorschrift für die Fehlerwahrscheinlichkeit gefunden werden. Diese Vorschrift konvergiert gegen einen Fixpunkt und gibt die best-mögliche Fehlerwarscheinlichkeit für einen Kode und ein Kanalmodell. Die Parameter α und β beeinflussen die Wahrscheinlichkeitsdichtefunktionen.

Die Berechnung der Wahrscheinlichkeitsdichten ist aber algorithmisch komplex und rechenintensiv. Oft gibt es keine geschlossenen Formeln für die Wahrscheinlichkeitsdichten. Es werden dann Histogramme aus Simulationen abgeleitet.

Da ich keine ~~Lust~~ Zeit hatte, *density evolution* für meine Simulationen zu implementieren oder vorhandene Implementation anzupassen, habe ich die Werte für die Parameter α und β per Suche bestimmt.

2.4.1 Bestimmen von α für NMS

Ich habe für jedes $\alpha \in [0, 1)$ mit der Schrittweite 0,1 die WER in Abhängigkeit von $\frac{E_b}{N_0}$ im Intervall von 0 dB bis 10 dB in 0,1 dB-Schritten simuliert. Dadurch konnte ich feststellen, dass ein optimales α für diese Kombination aus (31, 16, 7)-BCH-Kode und AWGN-Kanalmodell im Bereich von 0,8 bis 1 liegt.

Ich habe eine weitere Simulationsreihe von $\alpha \in [0{,}8, 1)$ mit der Schrittweite 0,08 gemacht. Die WER-Kurven liegen schon recht Nahe zusammen und sind im Diagramm nicht mehr auszuwerten.

Es sei ausserdem angemerkt, dass es für verschiedene $\frac{E_b}{N_0}$-Werte verschiedene α-Werte das beste Ergebnis haben. Um das optimale α zu finden, habe ich die Differenzen zwischen den WER-Kurven gebildet und die Differenzen aufsummiert. Diese sind in Abbildung 1 dargestellt.

Die Datenpunkte wurden mittels einer quadratischen Funktion auf $f(\alpha) = -12.70x^2 + 23.23x - 10.48$ approximiert. So kann α mit $\arg\max(f(\alpha)) = \frac{2323}{2540} \approx 0.914567$ bestimmt werden. Die interpolierende quadratische Funktion, sowie deren Maximum sind ebenfalls in Abbildung 1 visualisiert.

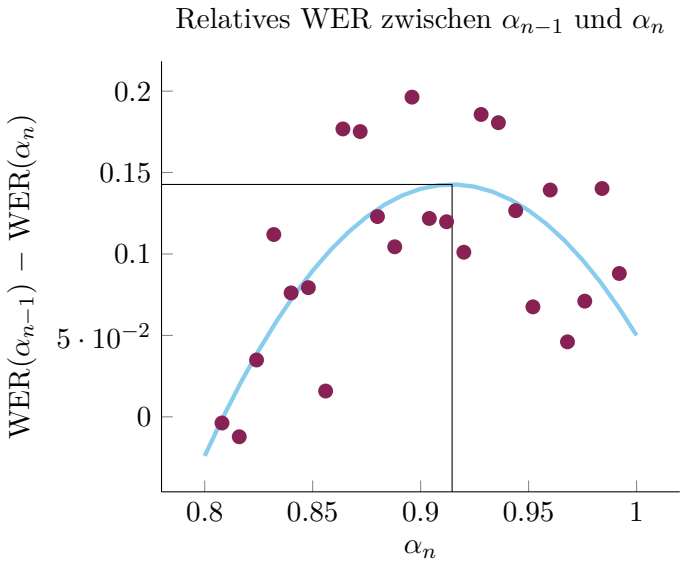

Abbildung 1: Relative Verbesserung, wenn α_{n-1} zu α_n geändert wird. Eine quadratische Funktion wurde approximiert um den Wert für α zu finden, der die WER minimiert.

Ich habe $\alpha = 0{,}915$ gewählt.

2.4.2 Bestimmung von β für OMS

Ich habe β in ähnlicher Weise wie den Parameter α bestimmt. Ich habe zuerst mit einer Schrittweite von 0,1 im Bereich $[0{,}1, 1)$ gesucht, um grafisch einen Überblick zu erhalten. Anschliessend habe ich den Bereich $\beta \in [-0{,}1, 0{,}1)$ in 0,01 Schritten genauer untersucht.

Anschließend wurde das gleiche Differenzverfahren mit Interpolation wie für die Bestimmung von α verwendet. Abbildung 2 stellt die Datenpunkte mit der quadratischen Interpolationsfunktion $f(x) = -108.176x^2 + 6.914x + 1.668$ dar. β wurde zu $\arg\max f(\beta) = \frac{3457}{108176} \approx 0{,}031957$ bestimmt.

Ich habe $\beta = 0{,}032$ gewählt.

2.4.3 Bestimmung von α und β 2D-NMS

Aufbauend auf der Bestimmung von α für NMS habe ich α und β für 2D-NMS im Bereich $[0{,}8, 1)$ in Schritten von 0,04 für ganzzahlige Werte von $\frac{E_b}{N_0} \in [0, 5]\,\mathrm{dB}$.

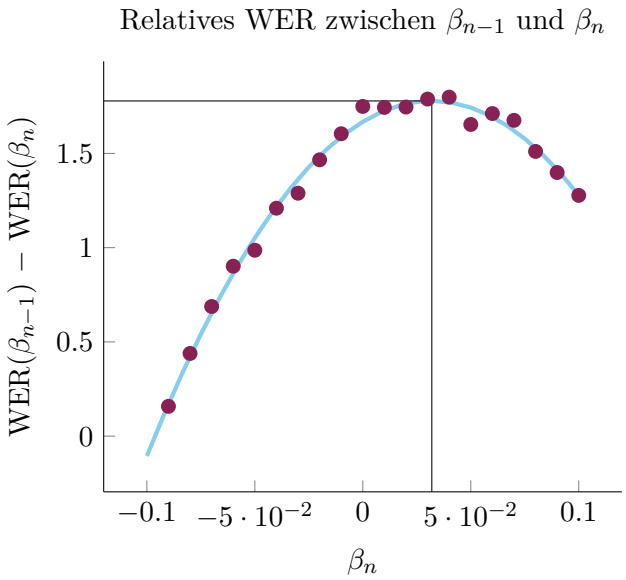

Abbildung 2: Relative Verbesserung, wenn β_{n-1} zu β_n geändert wird. Eine quadratische Funktion wurde approximiert um den Wert für β zu finden, der die WER minimiert.

Nachdem ich einen groben Überblick erhalten habe, habe ich die Simulation auf $\alpha \in [0{,}04, 0{,}99)$ und $\beta \in [0{,}9, 0{,}95)$ jeweils in Schritten von 0,01 eingeschränkt.

Für jeden Wert $\frac{E_b}{N_0}$ habe ich das α und β bestimmt, für die die simulierte WER minimal wird. Diese sind in Tabelle 1 abgedruckt. Abschließend habe ich die Parameter α und β aus deren arithmetischen Mittel über die verschiedenen $\frac{E_b}{N_0}$-Werte genommen.

Ich habe $\alpha = 0{,}968$ und $\beta = 0{,}907$ gewählt.

$\frac{E_b}{N_0}$ [dB]	α	β
0	0.97	0.91
1	0.99	0.91
2	0.97	0.9
3	0.96	0.9
4	0.98	0.92
5	0.94	0.9

Tabelle 1: α und β für das die WER von *2D-normalized Min-Sum* minimal wird.

2.5 Simulationen

Es wurden zwei Simulationen durchgeführt. Das erste Experiment simuliert einen bi-AWGN-Kanal mit in Abhängigkeit von $\frac{E_b}{N_0}$. Dieses Experiment soll die Leistungsfähigkeit der Dekodieralgorithmen für ein realistisches Kanalmodell vergleichbar machen.

Im zweiten Experiment werden in eine Kanalkodefolge eine definierte Anzahl harter Bitflips eingebracht. Dieses Experiment soll die unterschiedlichen Vorteile der Dekodierungsalgorithmen aufweisen. Während PGZ garantiert bis $w(e) \leq f_k$ korrigieren kann, haben iterative Dekodieralgorithmen diese Garantie nicht. Im Gegenzug können aber iterative Dekodieralgorithmen über f_k hinaus korrigieren, wobei bei PGZ in diesen Fällen entweder Dekodierversagen oder Falschkorrektur stattfindet.

2.5.1 Simulation Bi-AWGN-Kanal

Die *word error ratio* (WER) der Kodes wurde durch Simulation eines bi-AWGN-Kanal mit einem $\frac{E_b}{N_0}$ im Bereich 0 bis 10 dB in 0,1 dB-Schritten durchgeführt. Für die iterative Dekodierung wurde ein Limit 50 Iterationen gesetzt bevor Dekodierversagen erklärt wird. Nich betrachtet wurde der Einfluss der Anzahl der Iterationen auf die WER.

Die Anzahl der ausgewerteten Kodeworte ist an das WER der vorhergehenden Simulation gebunden. Für einen rauscharmen Kanal mit hohem SNR (also geringer WER) werden mehr Kodewörter ausgewertet um ein statistisch signifikantes Ergebnis zu erhalten. Bei niedrigem SNR ist die hohe WER schon bei vergleichsweise wenigen Wiederholungen stabil. Gleichzeitig kann so Simulationszeit gespart werden, da bei niedrigem SNR die iterative Dekodierung öfter in Dekodierversagen läuft. Dekodierung bei niedrigem SNR benötigt zwar weniger Iterationen, kann aber die exponentielle Abnahme des WER (und damit Zunahme von Simulationen) nicht ganz ausgleichen. Daher wird die Anzahl an Simulationen auf 10M beschränkt. Als Basis wurden 1.000 Wiederholungen bei einer WER von 1 gewählt. Wie in Gleichung 1 zu sehen, wird für jede Zehnerpotenz die die WER sinkt 10 mal so viele Simulationen durchgeführt.

$$N = \min \begin{cases} 1000 * \frac{1}{WER} \\ 10^7 \end{cases} \tag{1}$$

Die Simulationen wurden mit einem angenommenen WER von 0.5 gestartet.

2.5.2 Simulation BSC

Für dieses Experiment wurde $w(e)$ von 0 bis $2 * f_k$ variiert. Es wurden alle möglichen ($\sum_{x=0}^{6} \binom{31}{x} = 942649$) Fehlervektoren getestet. Für die iterativen Dekodierverfahren wird die Folge vor der Dekodierung von $0, 1^n$ nach $+1, -1^n$ abgebildet. Der iterative Dekodierer selbst rechnet mit Fließkommazahlen.

Für jedes $w(\mathbf{e})$ ist der Anteil and Fehlermustern als Prozentwert angegeben der eine Falschkorrektur oder Dekodierversagen provozierte.

Diese Simulation entspricht einem BSC mit einer Fehlerwahrscheinlichkeit von $p = \frac{w(\mathbf{e})}{n}$. Sie soll zeigen, ob und wenn ja wie weit mit iterativer Dekodierung für BCH-Kodes über f_k hinaus korrigiert werden kann.

2.6 Ergebnisse

Obwohl ich die WER für diskrete $\frac{E_b}{N_0}$-Werte simuliert habe, zeige ich in den Diagrammen kontinuierliche Linien. Dies hat zwei Gründe. Zum einen dienen die Linien der Übersichtlichkeit. Da ich die WER in 0,1 dB-Schritten simuliert habe, würden Marker sehr dich nebeneinander sitzen und einenander überdecken. Zum anderen werden durch das Verbinden von Punkten Trends besser hervorgehoben.

Aus zeitlichen Gründen habe ich keine Bestimmung von α und β für andere als den Anfangs gewählten (31, 16, 7)-BCH-Kode gemacht. Die ermittelten Parameter wurden auch für die Bestimmung des WER für andere Kodes benutzt.

2.6.1 Bi-AWGN-Kanal

Ich zeige die WER der simulierten Kodes in zwei Kombinationen. In der ersten Kombination zeige ich verschiedene Kodes der gleichen Länge dekodiert mit dem gleichen Algorithmus. Dies soll die Leistungsfähigkeit der verschiedenen Algorithmen in Abhängigkeit von der Kodestruktur darstellen. Abbildungen 3, 4 und 4 vergleichen BCH-Kodes der Länge 31 miteinander; Abbildungen 7 und 8 BCH-Kodes der Länge 63 und Abbildungen 11 und 12 BCH-Kodes der Länge 127.

Die zweite Kombination vergleicht die verschiedenen Algorithmen angewendet auf die gleichen BCH-Kodes. Damit soll die Leistungsfähigkeit der Algorithmen untereinander, sowie die Abhängigkeit der Leistungsfähigkeit von der Kodestruktur sichtbar gemacht werden. Abbildungen 5 und 6 vergleichen die Dekodieralgorithmen für BCH-Kodes der

Länge 31; Abbildungen 9 und 10 für BCH-Kodes der Länge 63 und Abbildungen 13 und 14 für BCH-Kodes der Länge 127.

Für den Vergleich der Dekodieralgorithmen für verschiedene BCH-Kodes in den Abildungen 3, 4, 7, 8, 11 und 12 wird als Referenz der Bereich grau hinterlegt in denen die unkodierte WER der entsprechenden Längen fallen. Für Abbildung 3 ist dies zum Beispiel die WER unkodierte Nachrichten der Länge 26 und 11.

Für den Vergleich der Dekodieralgorithmen für den gleichen BCH-Kode wird als Referenz die WER unkodierte Nachrichten der Länge l eingezeichnet.

BCH-Kodes der Länge 31 Die Übersicht über die *hard-decision*-Dekodierung mittels Berlekamp-Massey in Abbildung 3 zeigt nicht das erwartete Verhalten dass die WER mit zunehmenden Mindestabstand abnimmt.

Zum Beispiel hat der (31, 25, 4)-BCH-Kode eine höhere WER als der (31, 26, 3)-BCH-Kode. Erwarten würde man, dass diese Kodes gleichauf liegen, da beide ein Korrekturvermögen von $f_k = 3$ haben. Allerdings ist die simulierte Rauschvarianz abhängig von der Koderate, welche bei den Kodes verschieden ist. Daraus lässt sich schliessen das BCH-Kodes zusätzliche Redundanzstellen zumindest bei *hard-decision*-Dekodierung nicht effzient nutzen können.

Für den unmodifizierten *Min-Sum*-Algorithmus in Abbildung 4 zeigt sich das erwartete Bild einer abnehmenden WER mit zunehmenden Mindestabstand bis zum (31, 16, 7)-BCH-Kode. Kodes mit höherem Mindestabstand haben eine schlechtere Leistungsfähigkeit. Für *soft-decision*-Dekodierung ist unklar ob hier ebenfalls die zusätzliche Redundanz nicht genutzt werden kann, oder ob die Kodes mit größerem Mindesabstand eine ungeeignete Kontrollmatrix besitzen. Gleiches Verhalten zeigt sich für die Varianten *normalized Min-Sum*, *offset Min-Sum* sowie *self-correcting Min-Sum* in Abbildungen 4.

Der *self-correcting Min-Sum*-Algorithmus in der Variante 1 in Abbildung 3 weicht von der bisher beobachteten Leistungsfähigkeit ab. Die BCH-Kodes mit Mindestabstand 3 bis 6 zeigen die gleiche WER; ebenso liegen die Kodes mit den Mindestabständen 7 und 11 gleichauf. Der BCH-Kode mit dem Mindestabstand 8 ($l = 15$), liegt praktisch gleich auf mit der WER unkodierter Nachrichten der Länge 15. Dieser Algorithmus hat für diesen (31, 15, 8)-BCH-Kode praktisch keinen Kodierungsgewinn.

Generell zeigt (31, 16, 7)-BCH-Kode das beste Korrekturvermögen für iterative Dekodierung.

Die Abbildungen 5 und 6 vergleichen die WER-Kurven der untersuchten Algorithmen für verschiendene BCH-Kodes. Man sieht deutlich, dass die *hard-decision*-Algorithmen

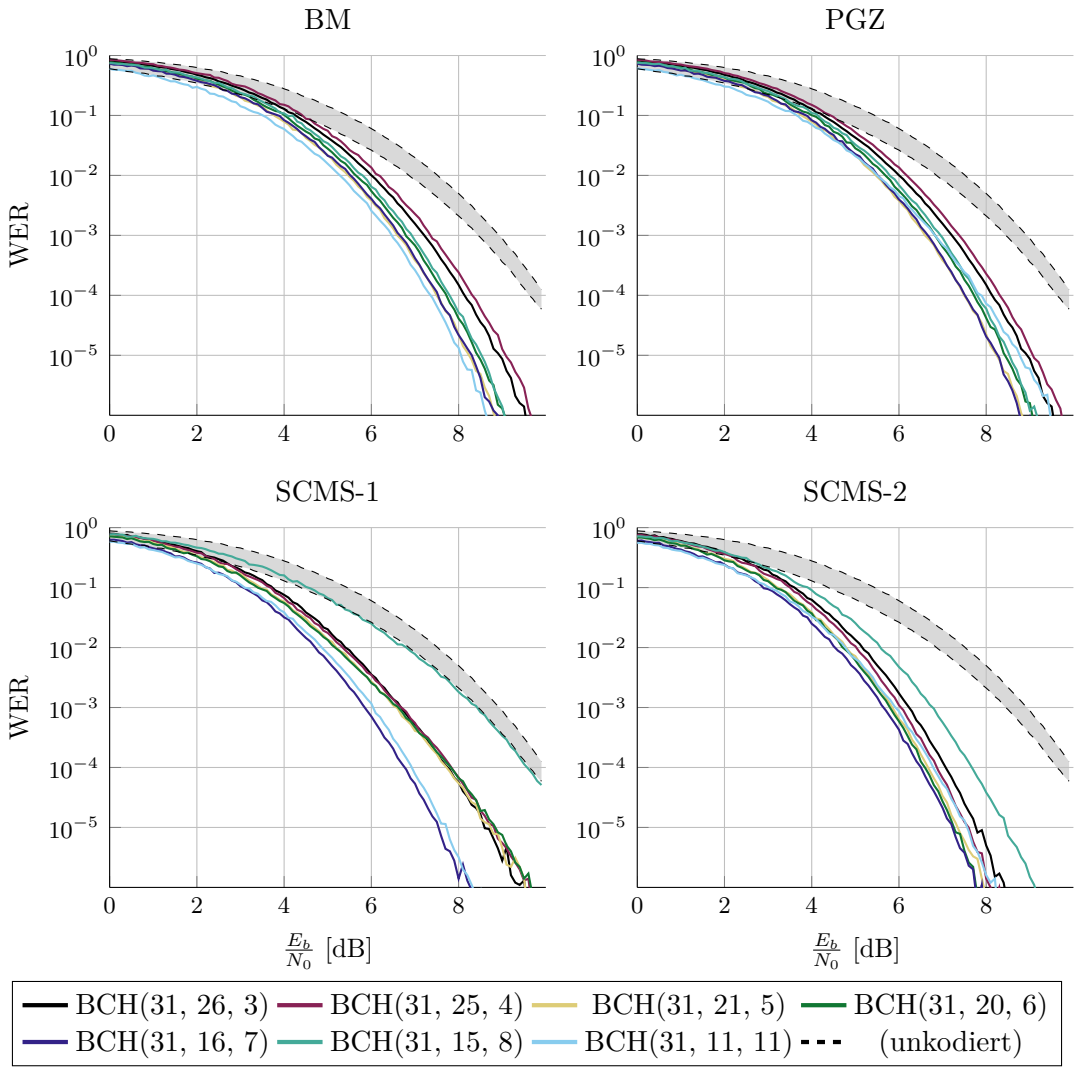

Abbildung 3: Übersicht über das Verhalten der Algorithmen PGZ, BM, SCMS-1 und SCMS-2, wenn für den BCH-Kode der Länge 31 der geplante Mindestabstand von 3 bis 9 variiert wird. Grau hinterlegt ist der Bereich in dem die WER unkodierter Wörter der Länge 11 bis 26 fallen.

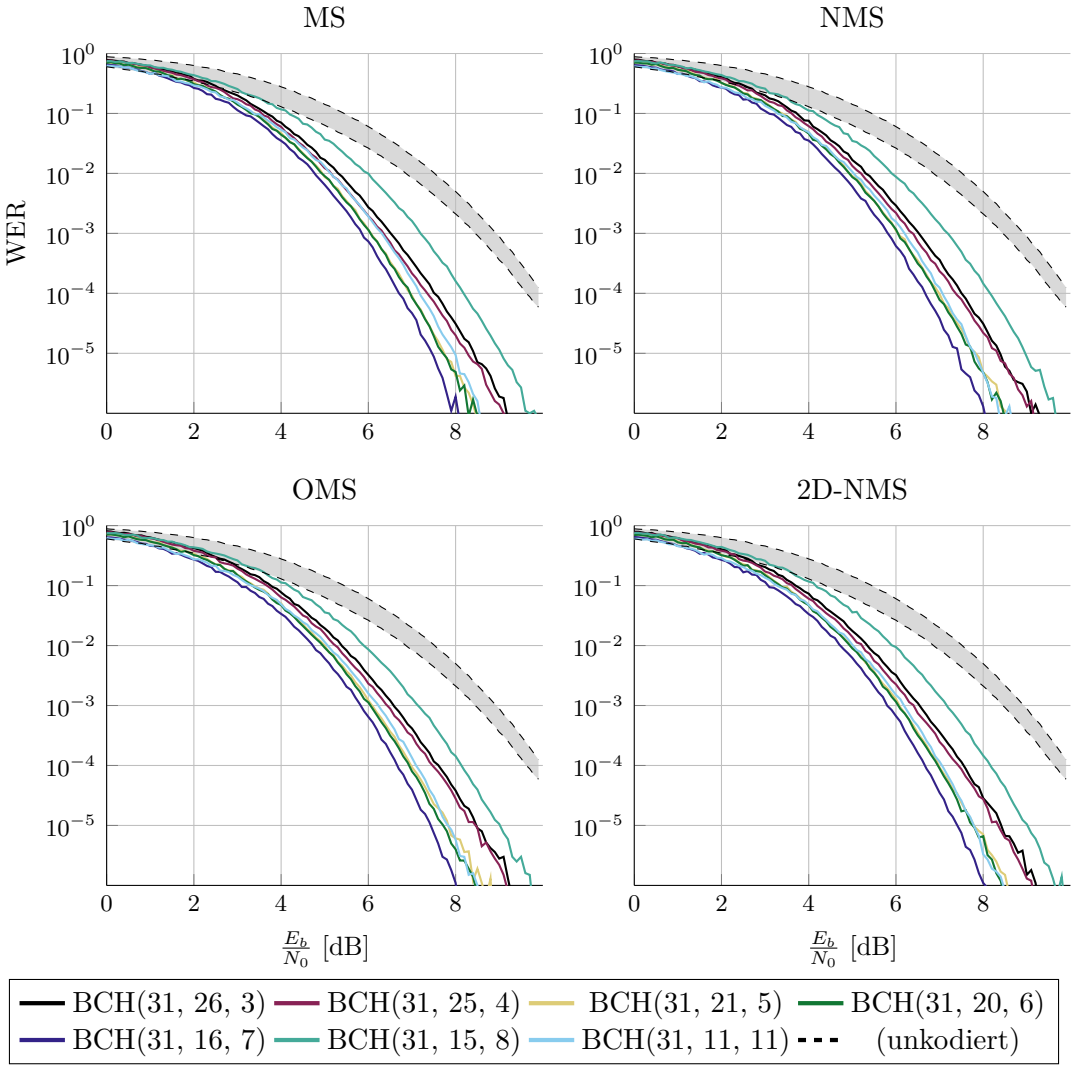

Abbildung 4: Übersicht über das Verhalten der Algorithmen MS, NMS, OMS und 2D-NMS wenn für den BCH-Kode der Länge 31 der geplante Mindestabstand von 3 bis 9 variiert wird. Grau hinterlegt ist der Bereich in den die WER unkodierter Wörter der Länge 11 bis 26 fallen.

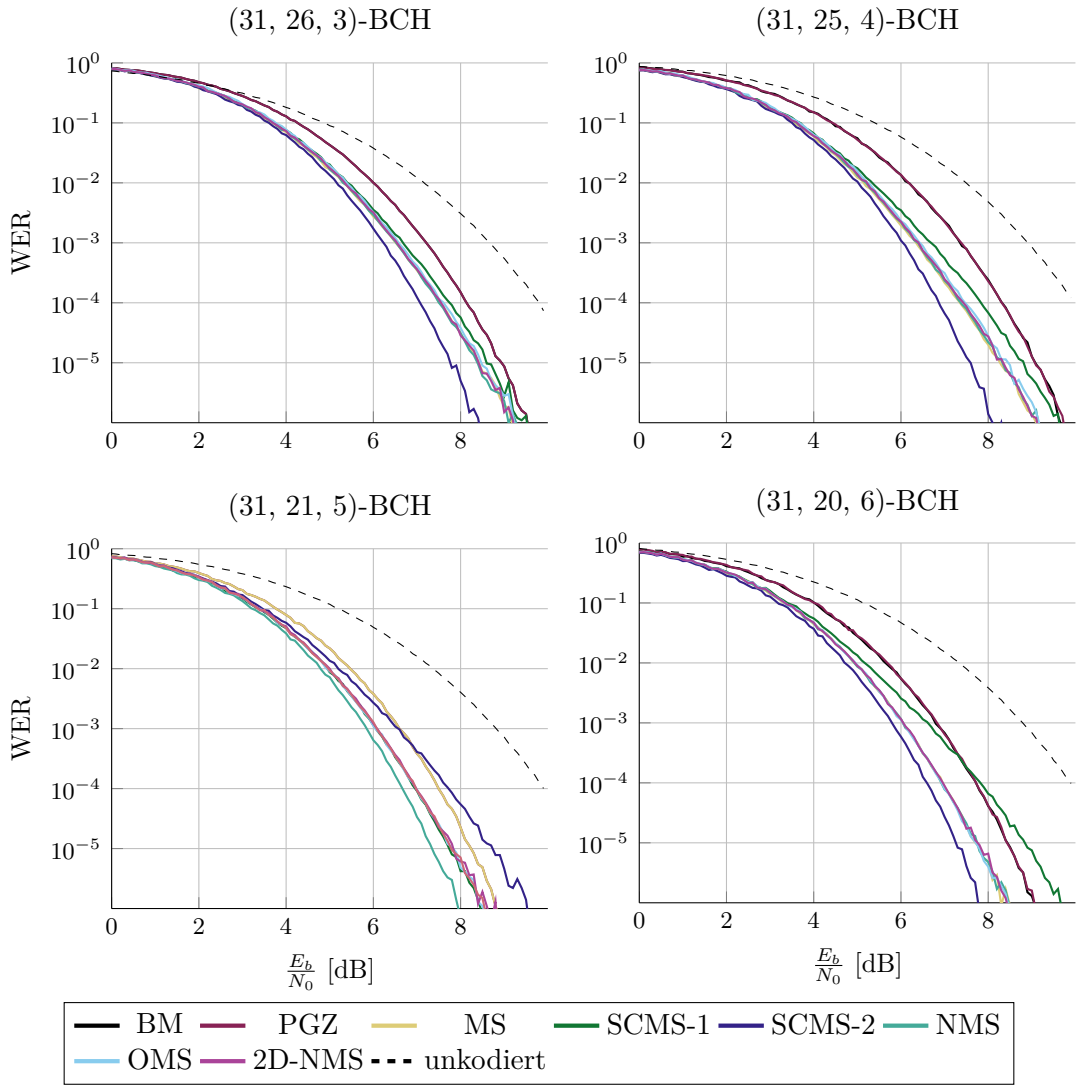

Abbildung 5: Wortfehlerverhältnisse für BCH-Kodes der Länge 31 und Mindestabstand von 3 bis 6 bei Korrektur mit PZG, BM, MS, NMS, OMS, 2D-NMS, SCMS-1 und SCMS-2. Als Referenz dient die WER unkodierter Nachrichten der Länge l des jeweiligen BCH-Kodes.

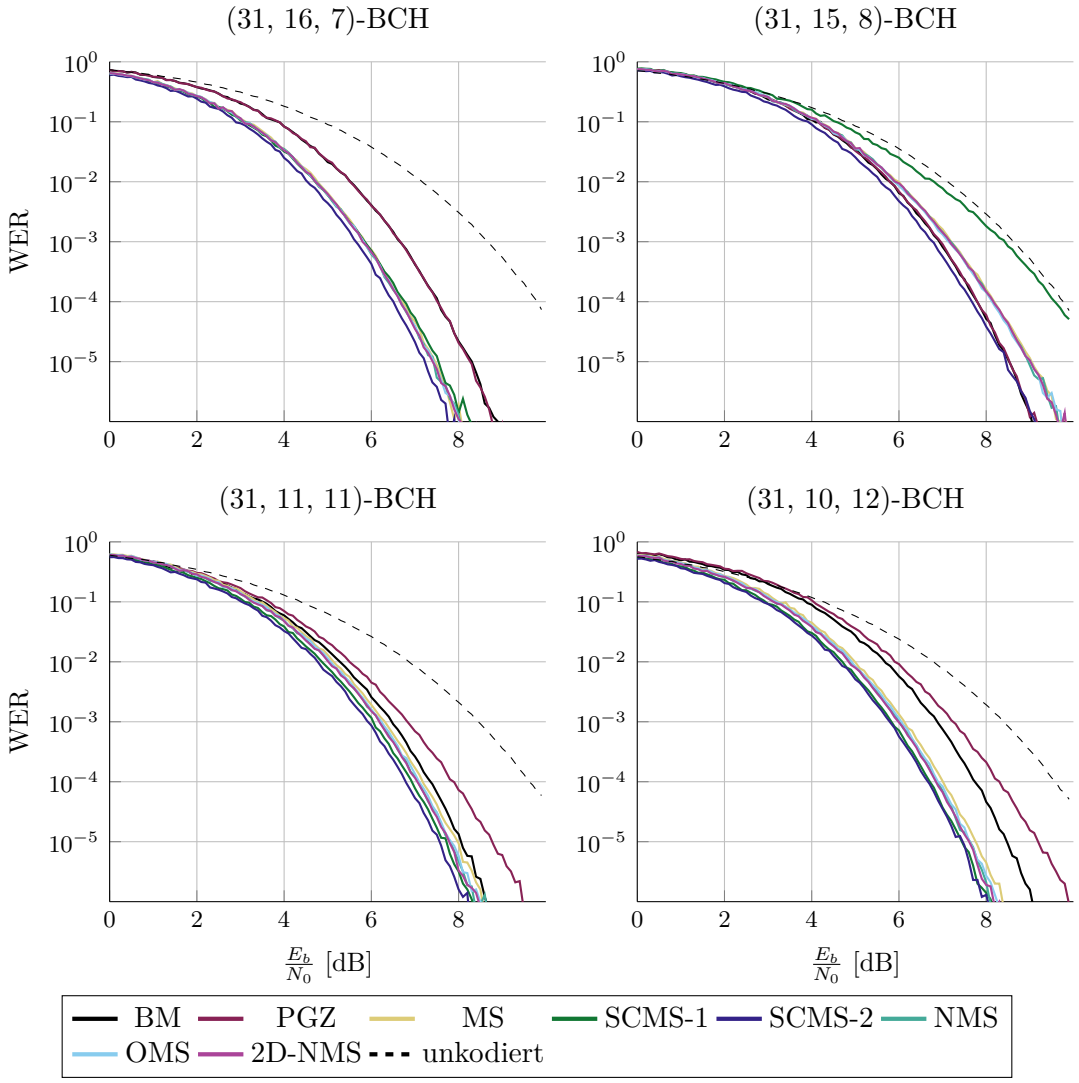

Abbildung 6: Wortfehlerverhältnisse für BCH-Kodes der Länge 31 und geplantem Mindestabstand von 7 bis 9 bei Korrektur mit PZG, BM, MS, NMS, OMS, 2D-NMS, SCMS-1 und SCMS-2. Als Referenz ist jeweils die WER unkodierter Nachrichten der Länge l des jeweiligen BCH-Kodes eingezeichnet.

BM und PGZ wesentlich schlechter ist als die iterativen *soft-decision*-Algorithmen abschneiden, ausgenommen der (31, 25, 5)-BCH-Kode.

Am besten schneidet der *self-correcting min-sum*-Algorithmus nach [10] ab.

NMS, OMS und 2D-NMS sind anscheinend nicht für diese Art von Kode geeignet. Zum Einen zeigte die Parameteroptimierung Werte für α und β Nahe dem neutralen Parameter für den die Algorithmen dem unmodifizierten *min-sum*-Algorithmus entsprechen (NMS: $\alpha = 1$, OMS: $\beta = 0$, 2D-NMS: $\alpha = \beta = 1$) . In [1] wurde für NMS $\alpha = 0{,}8$ und für OMS $\beta = 0{,}15$ bestimmt, allerdings für (8000, 4000)- und (1008, 504)-LDPC-Kodes. In [11] wurde für NMS $\alpha = 0{,}75$ und für 2D-NMS $\alpha = 0{,}83$, $\beta = 0{,}86$ für einen (16200, 7200)-LDPC-Kode ermittelt. Hier wäre es interessant zu sehen, welche Eigenschaften diese Algorithmen bei längeren BCH-Kodes zeigen.

Zum Anderen sind NMS, OMS und 2D-NMS nicht einmal für den optimierten (31, 16, 7)-BCH-Kode wesentlich besser als der unmodifizierte *min-sum*-Algorithmus.

Von den *sef-correcting*-Varianten ist SCMS-2 [10] besser as SCMS-1 [8]. Die Idee von *self-correcting Min-Sum* ist bei Änderung des Vorzeichens eines Variablenknotens ein neutrales Element zu setzen, da zumindest eine Paritätsgleichung für den Variablenkoten falsch ist. Der Hauptunterschied zwischen den Algorithmen ist die Wahl des neutralen Elementes. SCMS-1 setzt den Variablenknoten bei Vorzeichenänderung auf 0, SCMS-2 hingegen auf den Mittelwert von alten und neuem Wert. Letzteres scheint ausgesprochen vorteilhaft für BCH-Kodes zu sein, da SCMS-2 immer der Beste oder einer der besten Algorithmen für einen Kode ist. SCMS-1 hingegen degradiert für den (31, 15, 8)-BCH-Kode bis fast zur unkodierten WER.

Der unmodifiziertes *min-sum*-Algorithmus ist bis auf den (31, 15, 8)-BCH-Kode immer besser als PGZ oder BM.

BCH-Kodes der Länge 63 Abbildung 7 zeigt WER der untersuchten Algorithmen für BCH-Kodes der Länge 63 mit einem Mindestabstand von 3 bis 7; die Abbildung 8 mit einem Mindestabstand von 7 bis 11.

Für die *hard-decision*-Dekodierung mittels PGZ und BM zeigt sich wieder das gleiche Bild wie bei Kodes der Länge 31: Hat der Kode das Minimalpolynom m_0 als Faktor im Generatorpolynom trägt die zusätzliche redundante Stelle nicht zur Korrekturfähigkeit des Kodes bei. Die erhöhte Koderate bedingt aber eine höhere Rauschvarianz und damit verschlechtert sich die WER des Kodes.

Für die iterative *soft-decision*-Dekodierung mit *Min-Sum*-Algorithmus in Abbildung 8 zeigt sich vergleichbares Verhalten wie mit BCH-Kodes der Länge 31. Mit zunehmendem

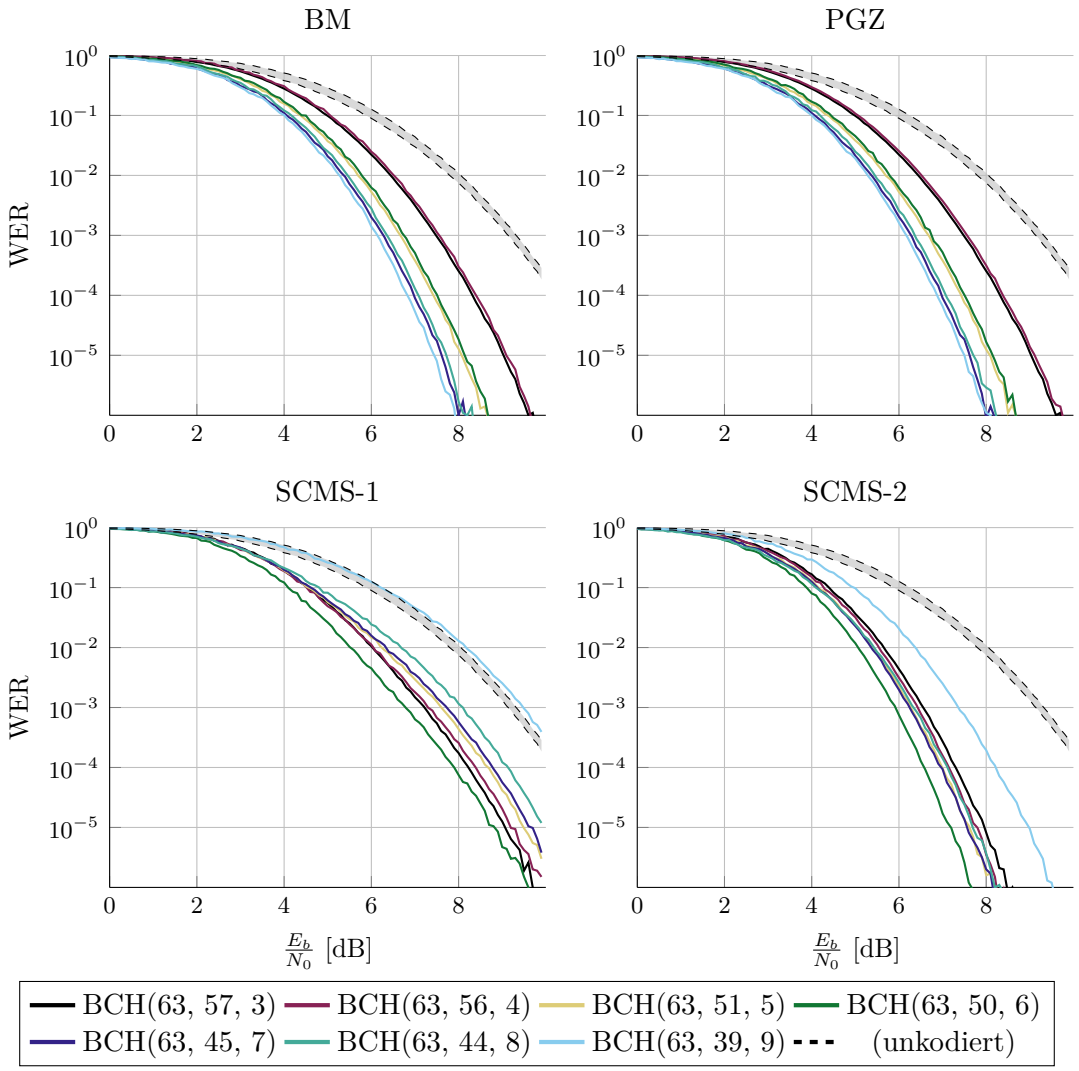

Abbildung 7: Übersicht über das Verhalten der Algorithmen PGZ, BM, SCMS-1 und SCMS-2, wenn für den BCH-Kode der Länge 63 der Mindestabstand von 3 bis 9 variiert wird. Als Referenz ist die WER unkodierter Nachrichten der Länge 39 bis 57, welche im grau schattierten Bereich liegen, eingezeichnet.

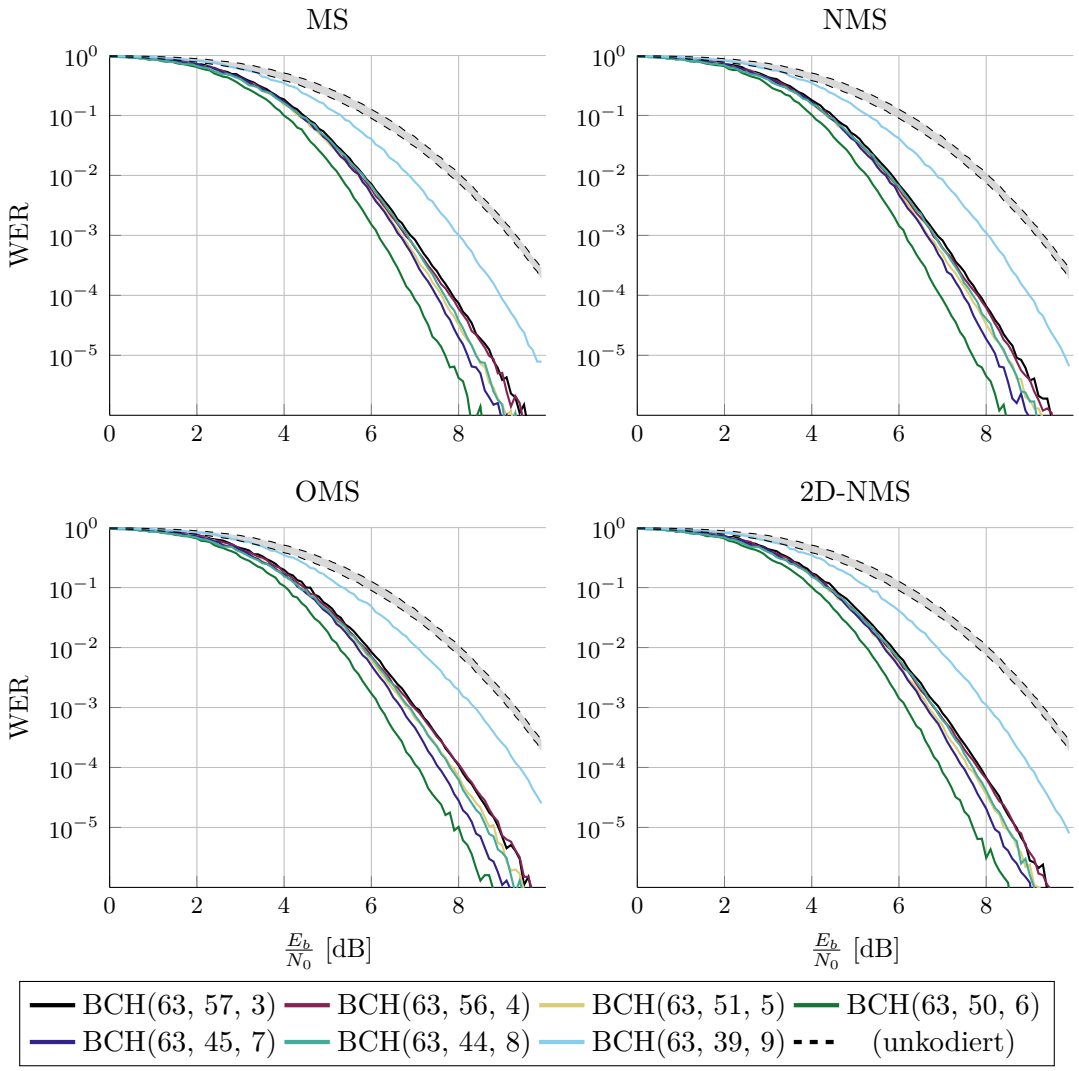

Abbildung 8: Übersicht über das Verhalten der Algorithmen MS, NMS, OMS und 2D-NMS wenn für den BCH-Kode der Länge 63 der Mindestabstand von 3 bis 9 variiert wird. Als Referenz dient die WER unkodierter Nachrichten der Länge 39 bis 57, welche im grau schattierten Bereich liegen.

Mindestabstand bis $d_{min} = 6$ verbessert sich die WER, danach verschlechtert sie sich wieder. Dieses Verhalten zeigt sich für alle iterativen Dekodieralgorithmen in Abbildunden 7 und 8.

Unabhängig vom Dekodierungsalgorithmus weist für iterative *soft-decision*-Dekodierung der (63, 50, 6)-BCH-Kode die geringste WER auf.

Im Gegensatz zu BCH-Kodes der Länge 31 sind die *hard-decision*-Dekodieralgorithmen nicht streng schlechter für BCH-Kodes der Länge 63.

Bis zum Mindestabstand von 6 hat der *self-correcting min-sum*-Algorithmus die kleinste WER. Danach liegt er für die Mindestabstände 7 und 8 gleich auf mit den *hard-decision*-Dekodieralgorithmen. Für einen Mindestabstand von 9 sind Berlekamp-Massey und PGZ besser als alle *soft-decision*-Algorithmen.

Die Reihenfolge unter den *soft-decision*-Algorithmen bleibt aber gleich. Unmodifiziertes *Min-Sum* liegt in etwa gleich auf mit *normalized*, *offset* und *2D-normalized Min-Sum*. SCMS-1 weist die schlechteste WER auf. Für den (63, 39, 9)-BCH-Kode ist dieser Dekodieralgorithmus sogar deutlich schlechter als unkodierte Nachrichten.

In Tabelle 2 ist das Gewicht des Kontrollpolynoms $h(x)$ für die untersuchten Kodes aufgelistet. Für die besonders geeigneten Kodes (31, 15, 8) und (63, 50, 6) lässt sich ein vergleichsweise niedriges Gewicht von $h(x)$ feststellen. Damit ist auch die Kontrollmatrix H dünner besetzt als für Kontrollpolynome mit größerem Gewicht. Da es aber auch Kodes mit geringerem oder gleichem Gewicht des Kontrollpolynoms gibt, zum Beispiel den (31, 11, 11)-BCH-Kode oder den (63, 44, 8)-BCH-Kode, die schlechtere Leistungsfähigkeit für iterative *soft-decision*-Dekodierung zeigen, lässt sich daraus schließen dass das Gewicht des Kontrollpolynoms nur ein Indiz für die Eignung eines BCH-Kodes für iterative Dekodierung ist.

BCH-Kodes der Länge 127 Für BCH-Kodes der Länge 127 setzt sich das Bild fort.

In den Abbildungen 13 und 14 ist die WER für BCH-Kodes der Länge 127 mit einem Mindestabstand von 3 bis 7 dargestellt.

Für die *hard-decision*-Algorithmen BM und PGZ wirkt sich das vorhandensein des Minimalpolynoms m_0 als Faktor im Generatorpolynom als eine Verschlechterung des WER aus. Dieses Phänomen nimmt allerdings mit zunehmender Länge des Kodes ab. Dies liegt daran das Koderatenunterschied bei Kodes mit m_0 und dem gleichen Kode ohne m_0 im Generatorpolynom äußerst gering ausfällt.

Für *soft-decision*-Dekodierung verbessert sich die Leistungsfähigkeit zuerst mit zuneh-

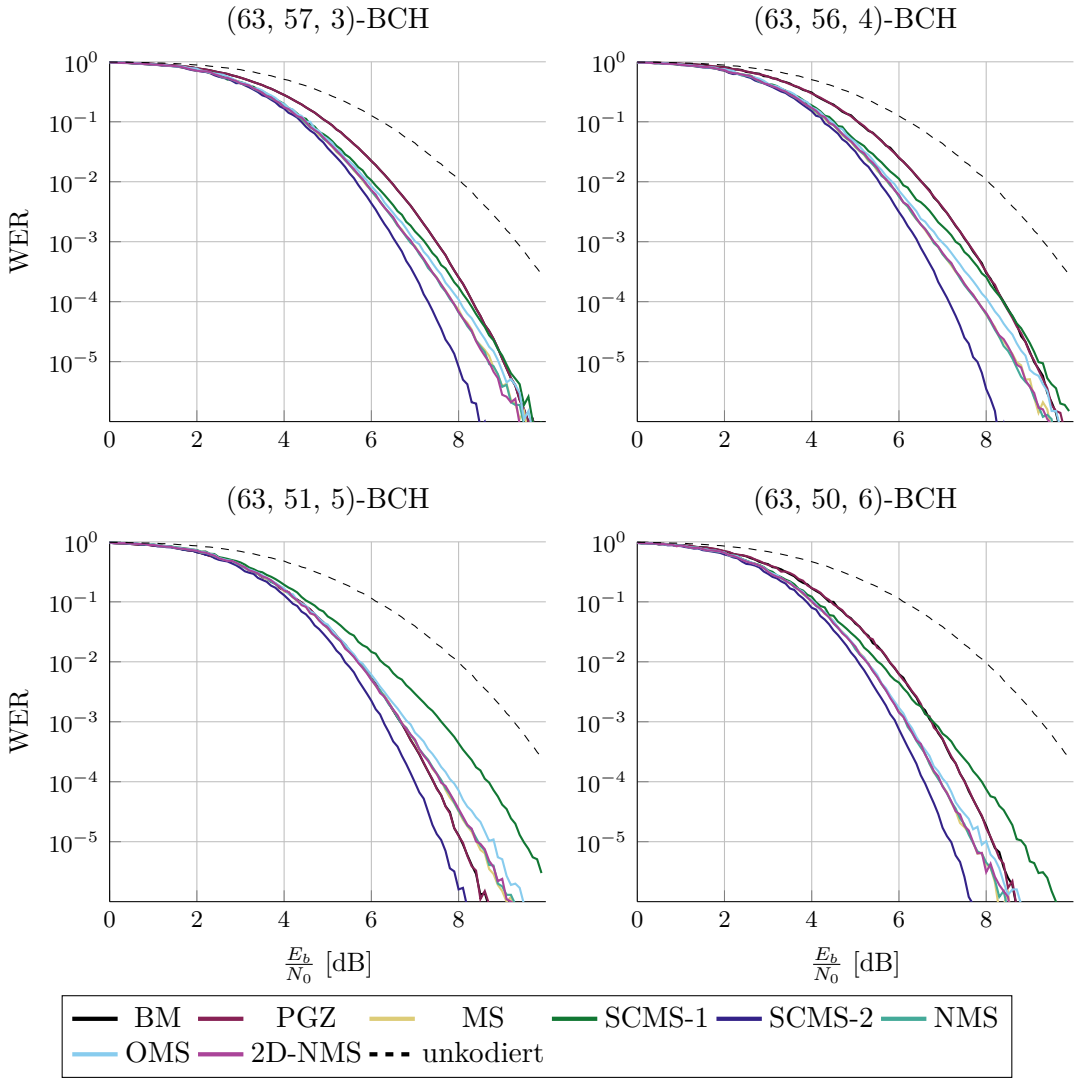

Abbildung 9: Wortfehlerverhältnisse für BCH-Kodes der Länge 63 und Mindestabstand von 3 bis 6 bei Korrektur mit PZG, BM, MS, NMS, OMS, 2D-NMS, SCMS-1 und SCMS-2. Als Referenz dienen jeweils unkodierte Nachrichten der Länge l der jeweiligen Kodes.

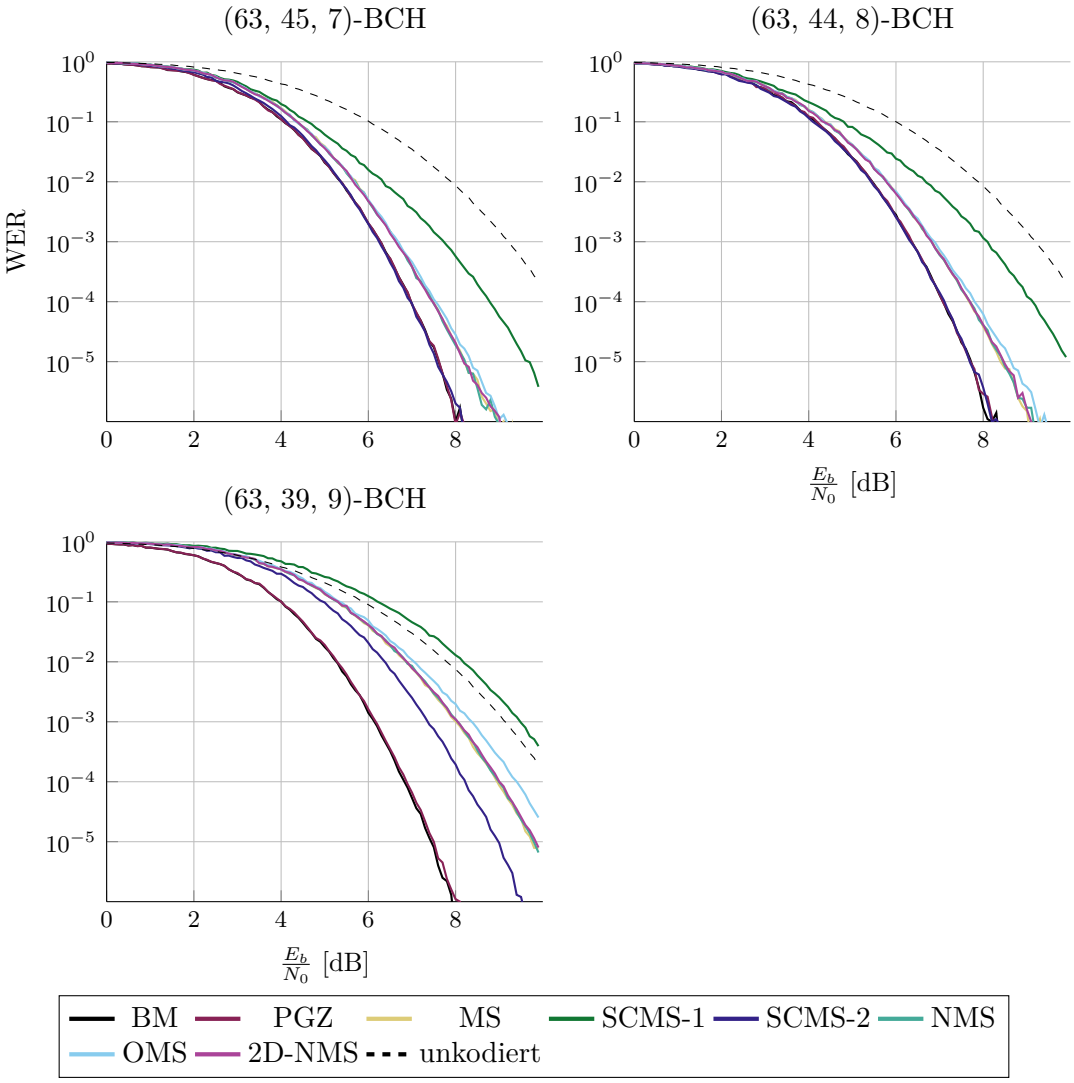

Abbildung 10: Wortfehlerverhältnisse für BCH-Kodes der Länge 63 und Mindestabstand von 7 bis 9 bei Korrektur mit PZG, BM, MS, NMS, OMS, 2D-NMS, SCMS-1 und SCMS-2. Als Referenz ist die WER unkodierter Nachrichten der Länge l des entsprechenden BCH-Kodes aufgetragen.

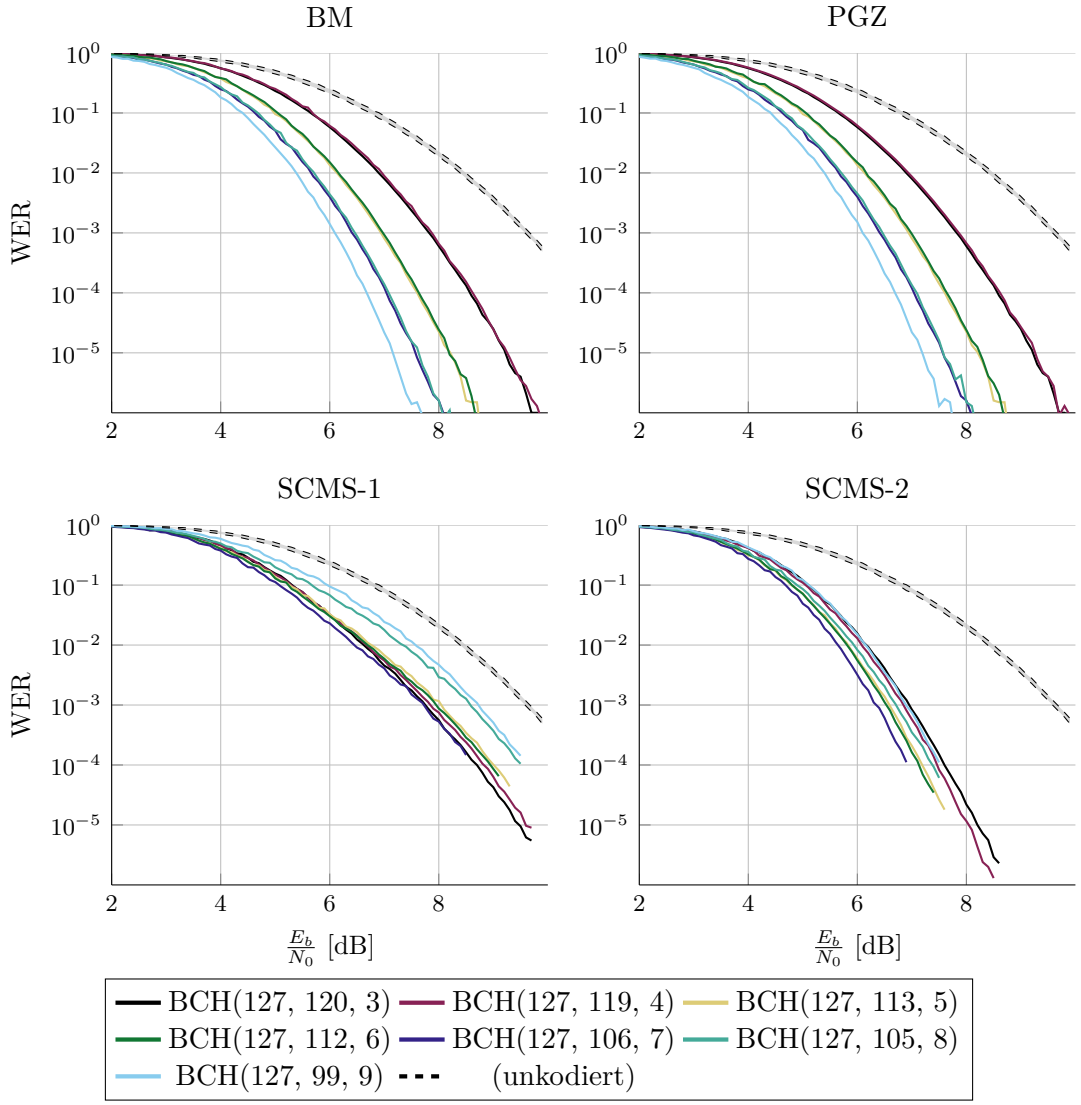

Abbildung 11: Übersicht über das Verhalten der Algorithmen PGZ, BM, SCMS-1 und SCMS-2, wenn für den BCH-Kode der Länge 127 der geplante Mindestabstand von 3 bis 9 variiert wird. Als Referenz ist die WER unkodierter Nachrichten der Länge 99 bis 120, welche im grau schattierten Bereich liegen, eingezeichnet.

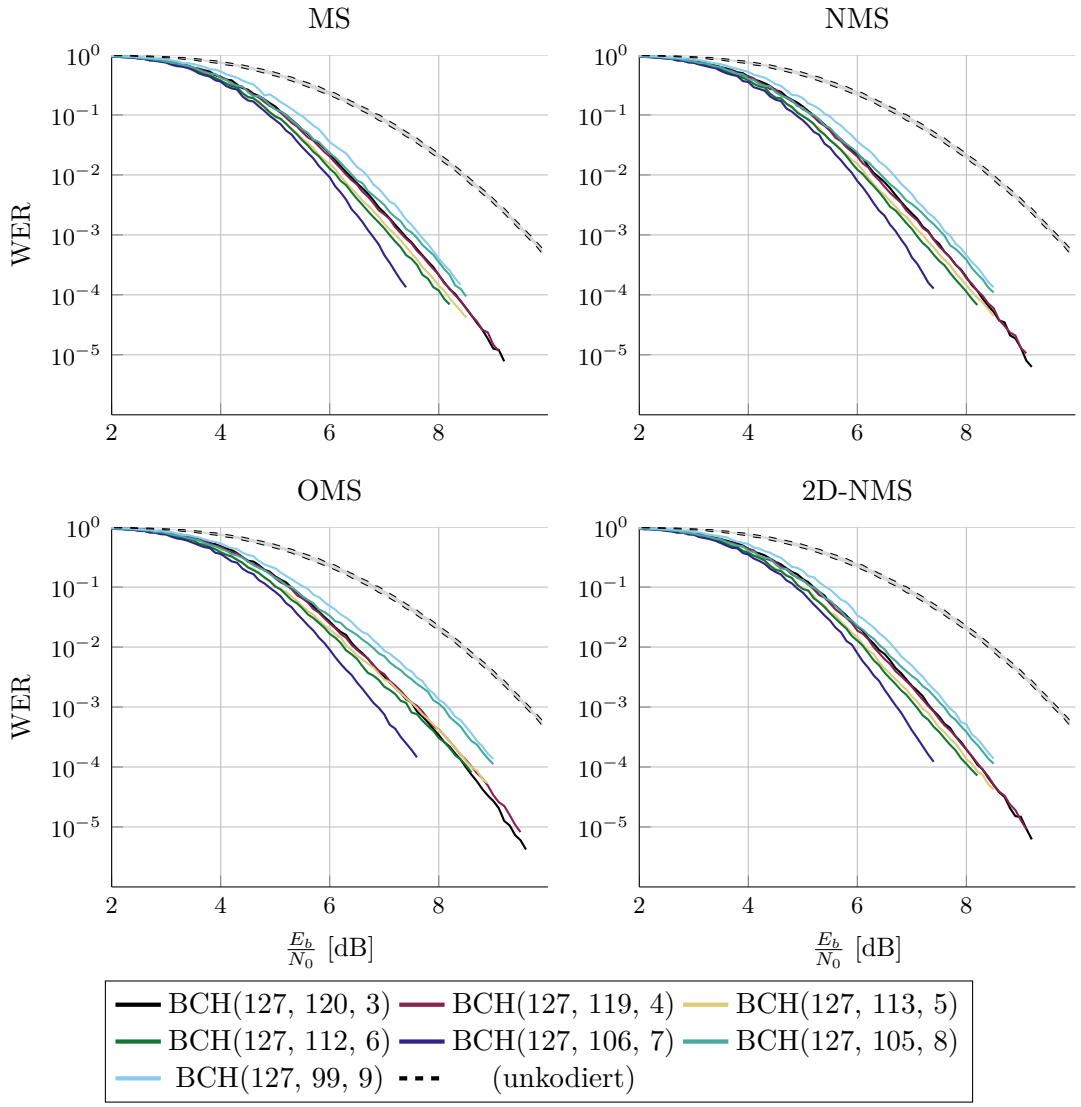

Abbildung 12: Übersicht über das Verhalten der Algorithmen MS, NMS, OMS und 2D-NMS wenn für den BCH-Kode der Länge 127 der geplante Mindestabstand von 3 bis 9 variiert wird. Als Referenz dient die WER unkodierter Nachrichten der Länge 99 bis 120, welche im grau schattierten Bereich liegen.

menden Mindestabstand, bis sie schließlich für Mindestabstände größer 6 wieder sinkt.

Der (127,106, 7)-BCH-Kode ist der am Besten für iterative Dekodierung geeignetste BCH-Kode der Länge 127.

Wie für BCH-Kodes der 63 schon beobachtet sind auch für BCH-Kodes der Länge 127 die *hard-decision*-Dekodieralgorithmen BM und PGZ nicht immer schlechter als iterative *soft-decision*-Dekodierung. Für $d_{min} = 7$ liegt BM und PGZ mit dem besten *soft-decision*-Algorithmus *self-correcting Min-Sum* in der Variante 2 gleich auf. Bei höherem Mindestabstand sind die *hard-decision*-Algorithmen sogar besser.

BCH-Kodes der Länge 15 Um einen besseren Überblick zu bekommen, habe ich mich entschieden weitere BCH-Kodes zu untersuchen. Da für zunehmende Länge die Simulationszeit zunimmt, habe ich Kodes der Länge 15 untersucht.

In den Abbildungen 15 und 16 werden die Dekodieralgorithmen für BCH-Kodes der Länge 15 mit Mindestabständen von 3 bis 15 untersucht.

Mit iterativer *soft-decision*-Dekodierung zeigt sich der (15, 7, 5)-BCH-Kode am leistungsfähigsten. Für *hard-decision*-Dekodierung liegen die Kodes (15, 11, 3) und (15, 5, 7) mit dem (15, 7, 5)-BCH-Kode ungefähr gleich auf.

Der (15, 1, 15)-BCH-Kode ist für BM und PGZ schlechter als unkodierte Nachrichten der Länge 1. Für iterative *soft-decision*-Dekodierung liegt der (15, 1 15)-BCH-Kode mit der BER gleich auf. Nur NMS und 2D-NMS weichen davon ab; OMS weicht nur gering ab. Zu erklären ist das mit der Parametrisierung dieser Algorithmen für den (31, 16, 7)-BCH-Kode.

In den Abbildungen 17 und 18 werden die Dekodierungsalgorithmen für den gleichen Kode gegenübergestellt.

Für die Mindestabstände 3 und 4 ist SCMS-2 am Besten. Die restlichen iterativen *soft-decision*-Algorithmen liegen gleich auf. *Hard-decision*-Dekodierung mit BM und PGZ ist deutlich schlechter.

Für die Mindestabstände 5 bis 8 liegen alle *soft-decision*-Algorithmen in etwa gleich auf, vor BM und PGZ. Für den (15, 1, 15)-BCH-Kode erreicht kein Algorithmus Kodierungsgewinn über unkodierte Nachrichten der Länge 1.

Einfluss der Kodeparameter Ich habe die Leistungsfähigkeit iterativer *soft-decision*-Dekodieralgorithmen für BCH-Kodes verschiedener Länge und Mindestabstände simu-

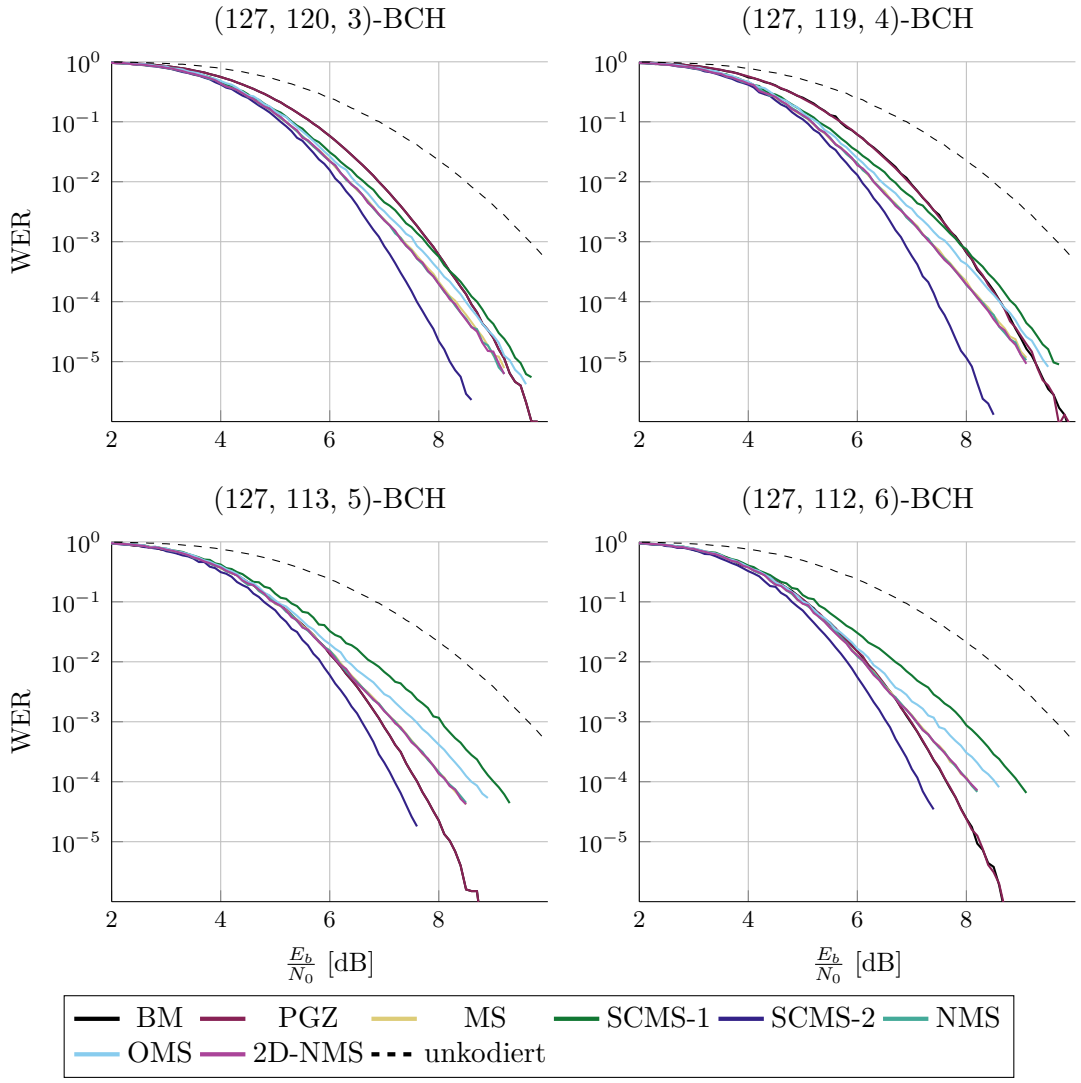

Abbildung 13: Wortfehlerverhältnisse für BCH-Kodes der Länge 127 und Mindestabstand von 3 bis 6 bei Korrektur mit PZG, BM, MS, NMS, OMS, 2D-NMS, SCMS-1 und SCMS-2. Als Referenz dienen jeweils unkodierte Nachrichten der Länge l der jeweiligen Kodes.

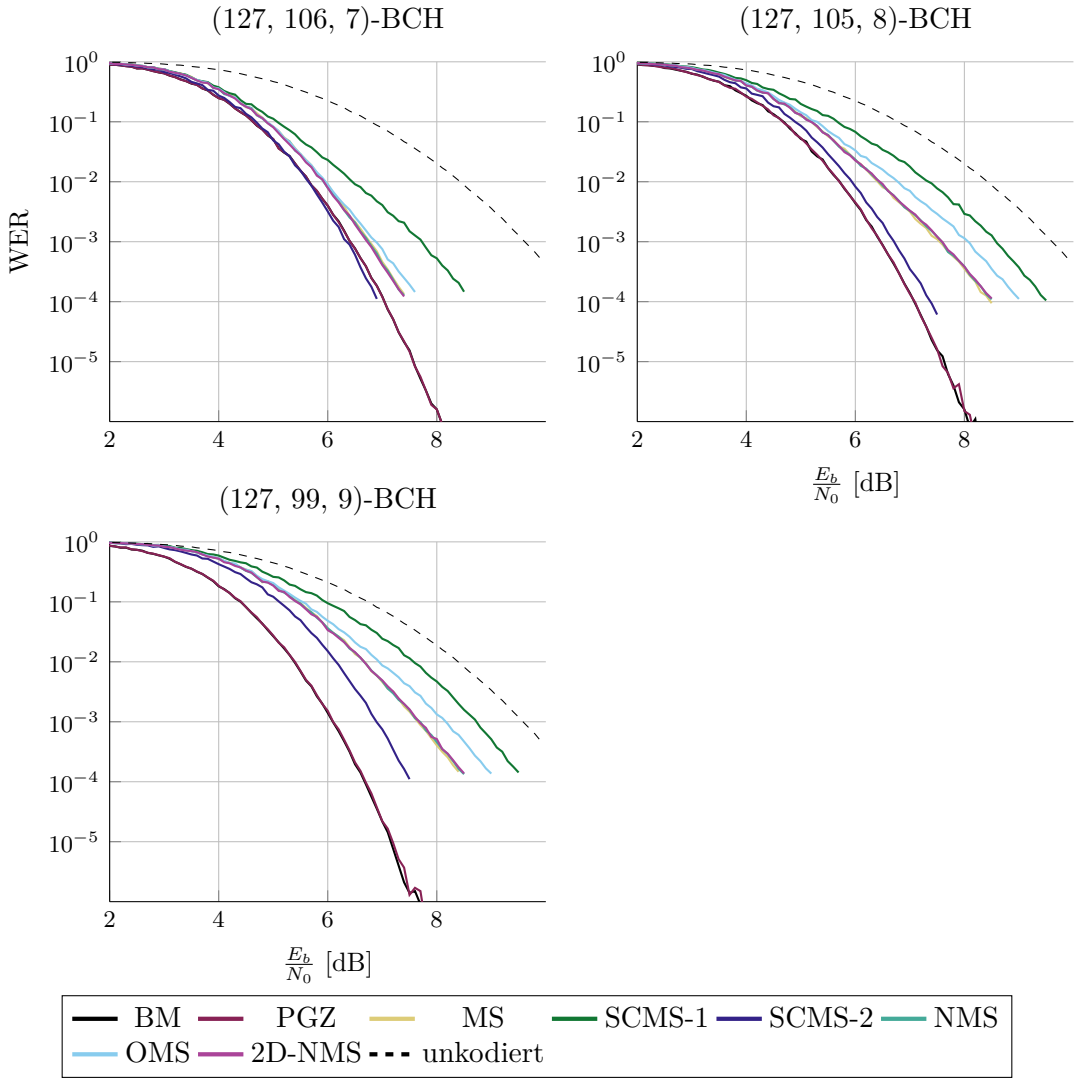

Abbildung 14: Wortfehlerverhältnisse für BCH-Kodes der Länge 127 und geplantem Mindestabstand von 7 bis 9 bei Korrektur mit PZG, BM, MS, NMS, OMS, 2D-NMS, SCMS-1 und SCMS-2. Als Referenz ist die WER unkodierter Nachrichten der Länge l des entsprechenden BCH-Kodes aufgetragen.

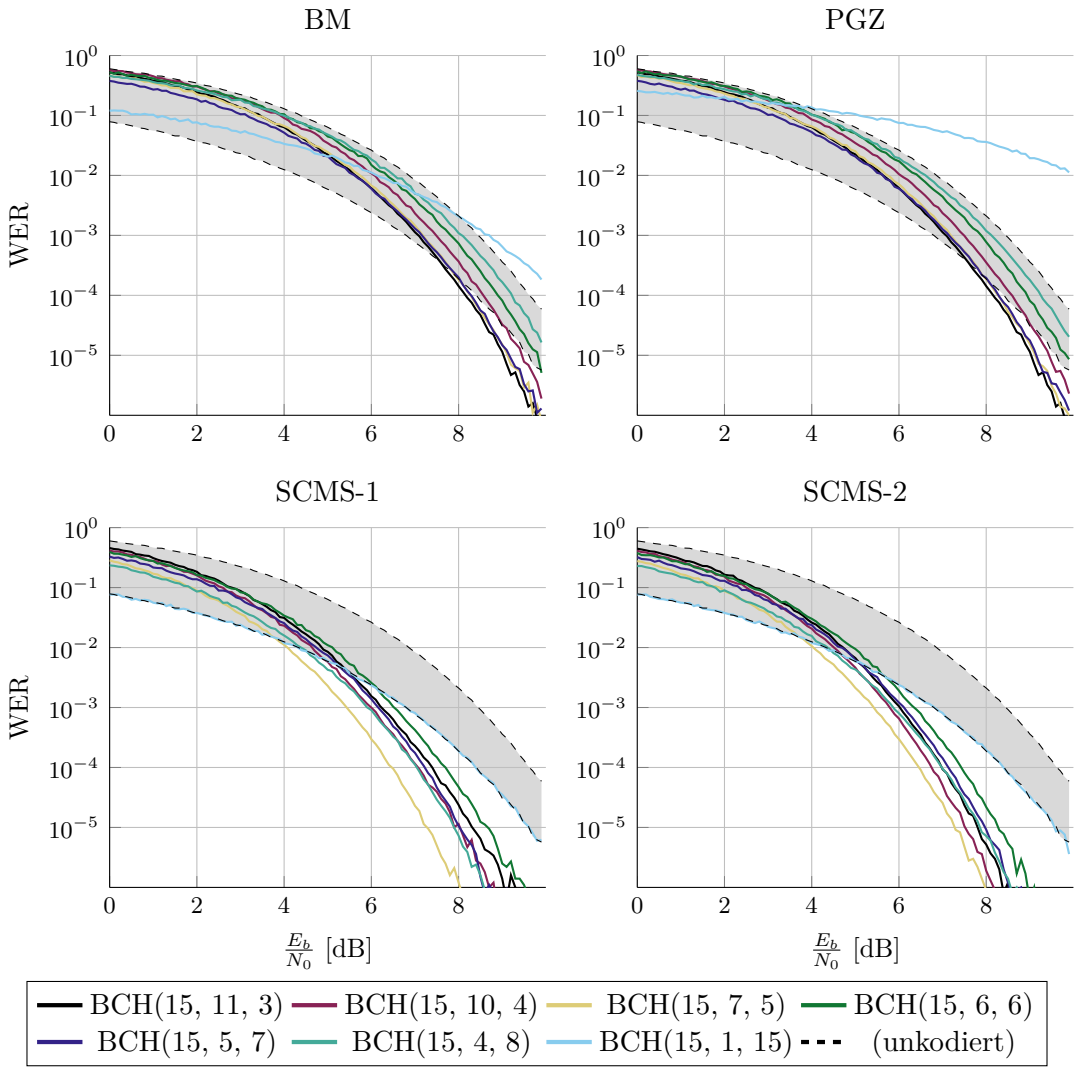

Abbildung 15: Übersicht über das Verhalten der Algorithmen PGZ, BM, SCMS-1 und SCMS-2, wenn für den BCH-Kode der Länge 15 der Mindestabstand von 3 bis 9 variiert wird. Als Referenz ist die WER unkodierter Nachrichten der Länge 1 bis 11, welche im grau schattierten Bereich liegen, eingezeichnet.

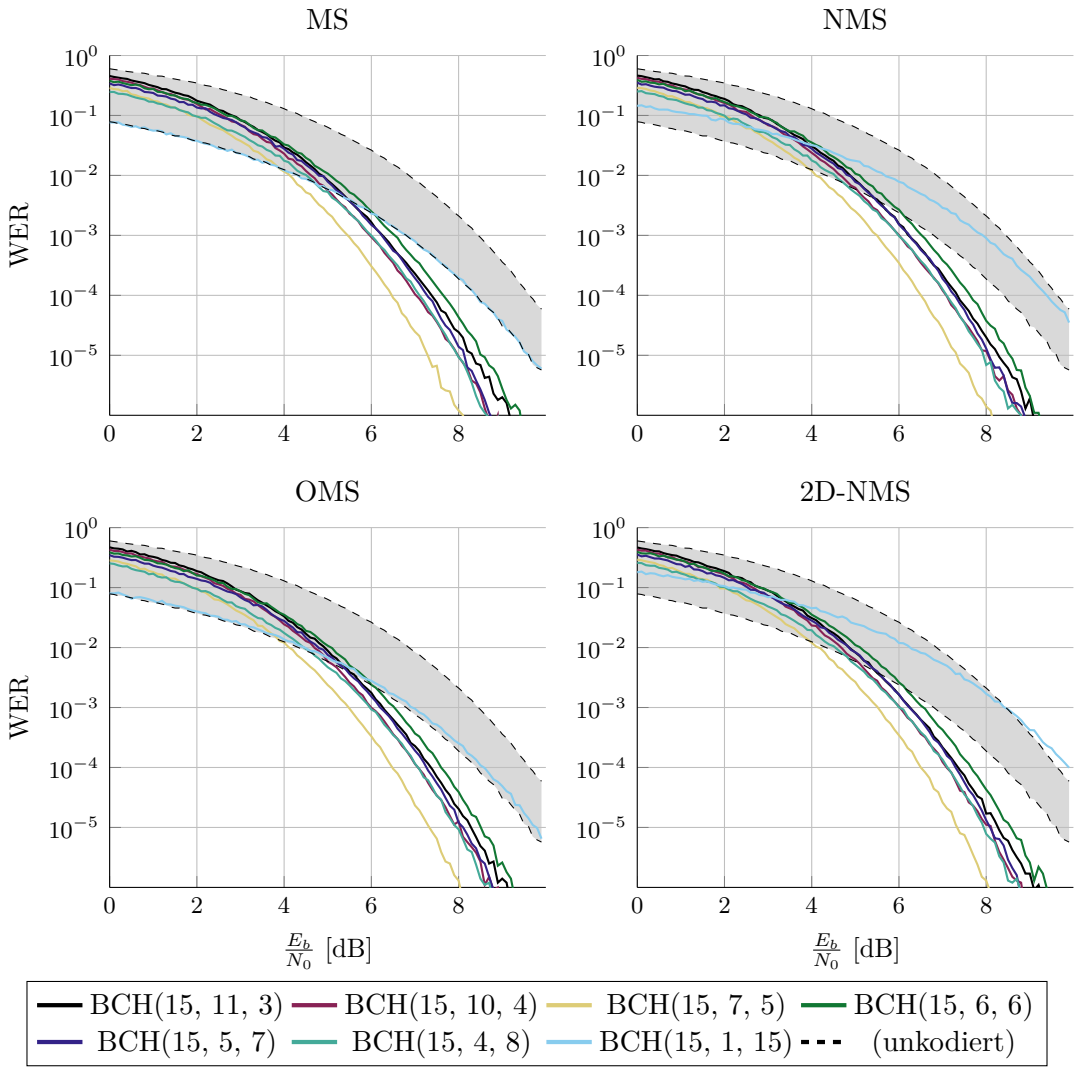

Abbildung 16: Übersicht über das Verhalten der Algorithmen MS, NMS, OMS und 2D-NMS wenn für den BCH-Kode der Länge 15 der Mindestabstand von 3 bis 8 variiert wird. Als Referenz dient die WER unkodierter Nachrichten der Länge 1 bis 11, welche im grau schattierten Bereich liegen.

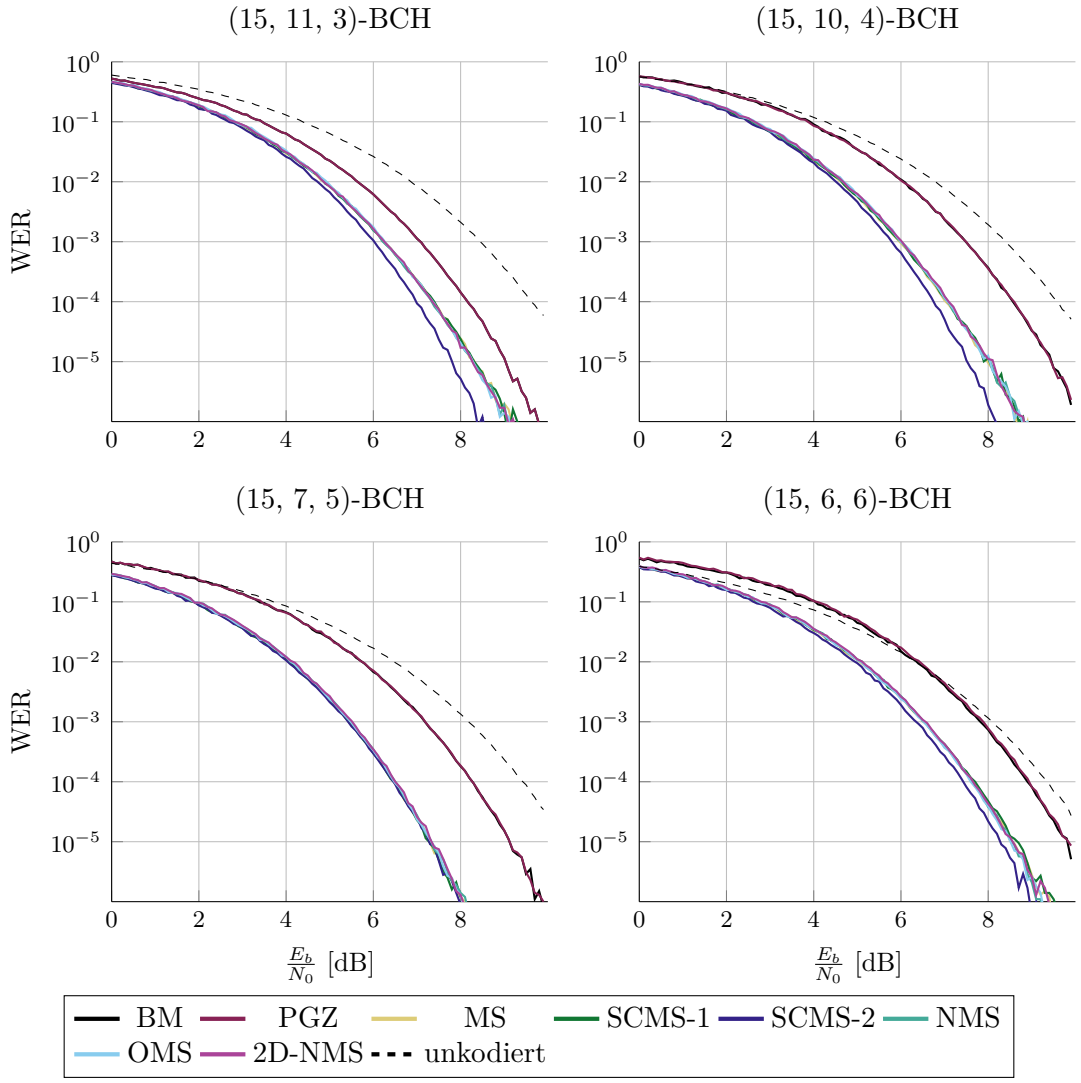

Abbildung 17: Wortfehlerverhältnisse für BCH-Kodes der Länge 15 und Mindestabstand von 3 bis 6 bei Korrektur mit PZG, BM, MS, NMS, OMS, 2D-NMS, SCMS-1 und SCMS-2. Als Referenz dienen jeweils unkodierte Nachrichten der Länge l der jeweiligen Kodes.

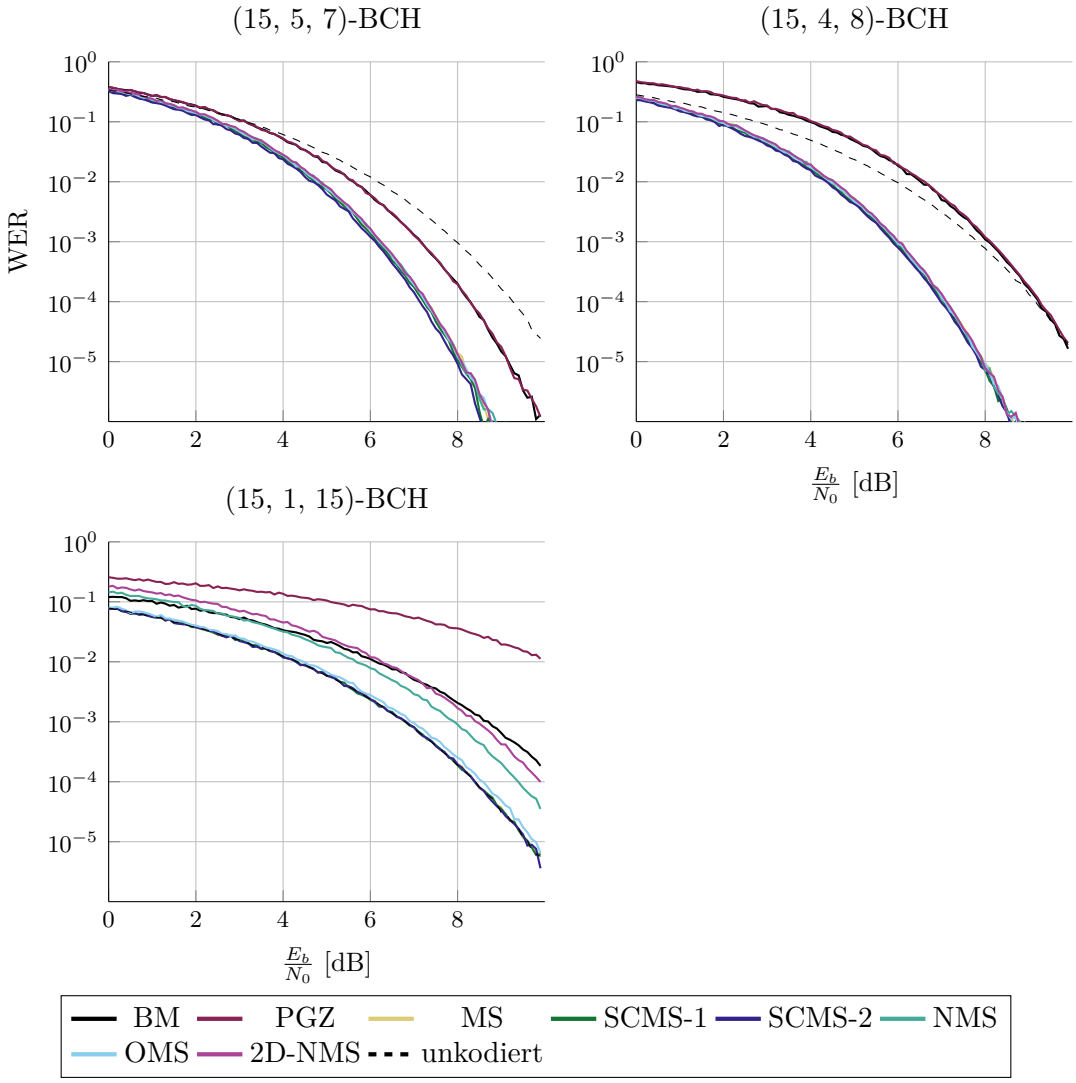

Abbildung 18: Wortfehlerverhältnisse für BCH-Kodes der Länge 15 und geplantem Mindestabstand von 7 bis 9 bei Korrektur mit PZG, BM, MS, NMS, OMS, 2D-NMS, SCMS-1 und SCMS-2. Als Referenz ist die WER unkodierter Nachrichten der Länge l des entsprechenden BCH-Kodes aufgetragen.

liert und mit den klassischen *hard-decision*-Dekodieralgorithmen BM und PGZ verglichen. Dabei hat sich der *self-correcting min-sum*-Algorithmus in der Variante 2 als leistungsfähigster *soft-decision*-Algorithmus für BCH-Kodes herauskristallisiert. Wie erwartet hängt die Leistungsfähigkeit der iterativen *soft-decision*-Dekodieralgorithmen aber stark von der Kodestruktur ab.

Nun stellt sich die Frage welche BCH-Kodes sich besonders für *soft-decision*-Dekodierung eignen. Dazu sind in Tabelle 2 die Kodeparameter aller untersuchter Kodes aufgelistet. Für jede Länge sind die BCH-Kodes für die das kleinste WER in Verbindung mit iterativer *soft-decision*-Dekodierung ermittelt wurde farbig hervorgehoben.

Die Kodes fallen mit einem ähnlichen Mindestabstand d_{min} von 6 oder 7 auf.

Die Koderate liegt bei 0,5 für den (31, 16, 7)-BCH-Kode und $\approx 0{,}8$ für die (63, 50, 6)-BCH und (127, 106, 7)-BCH-Kodes. Ausserdem wurde der (63, 30, 13)-BCH-Kode mit *self-correcting Min-Sum* Variante 2 untersucht. Die WER-Kurve dieses Kodes ist nicht dargestellt, aber sehr viel schlechter als für den (63, 50, 6)-BCH-Kode. Daraus kann man schließen, dass die Koderate keinen Einfluss auf die Eignung eines BCH-Kodes für iterative *soft-decision*-Dekodierung hat.

Da die iterativen Algorithmen für die Dekodierung von LDPC-Kodes benutzt werden ist davon auszugehen, dass das Gewicht der Kontrollmatrix und damit des Kontrollpolynoms Einfluss auf die Leistungsfähigkeit eines iterativen *soft-decision*-Algorithmus hat. In 2 ist daher auch das Gewicht des Kontrollpolynoms abgedruckt. Für Kodes der Länge 31 weist der beste Kode nur das zweit kleinste Gewicht auf. Für Kodes der Länge 63 hat der beste Kode zwar das kleinste Gewicht aber ein zweiter Kode mit gleichem Gewicht hat sich als wesentlich ungeeigneter für iterative *soft-decision*-Dekodierung herausgestellt. Für Kodes der Länge 127 hat der Beste Kode tatsächlich das kleinste Gewicht.

In Abbildung 19 sind die WER-Kurven der geeignetsten Kodes für den SCMS-2 Algorithmus zusammen dargestellt. Hier sieht man vergleichend, dass mit zunehmendem Gewicht der Kodes die WER schlechter wird.

Unklar ist, ob sich diese Beobachtungen verallgemeinern lassen. Ein Trend ist hingegen erkennbar. Geeignete Kodes scheinen einen Mindestabstand von 6 oder 7 zu haben. Am entscheidensten scheint aber das Gewicht des Kontrollpolynoms, welches möglichst gering sein sollte. Abbildung 19 zeigt zudem auch, dass bis $\frac{E_b}{N_0} \approx 7\,\mathrm{dB}$ kürzere Kodes zu bevorzugen sind.

BCH-Kode	Koderate	$w(h(x))$
(15, 11, 3)	0,73	8
(15, 10, 4)	0,67	7
(15, 7, 5)	0,47	4
(15, 6, 6)	0,40	5
(15, 5, 7)	0,33	4
(15, 4, 8)	0,27	3
(15, 1, 15)	0,07	2
(31, 26, 3)	0,84	16
(31, 25, 4)	0,81	15
(31, 21, 5)	0,68	12
(31, 20, 6)	0,65	11
(31, 16, 7)	0,52	8
(31, 15, 8)	0,48	11
(31, 11, 11)	0,35	6
(63, 57, 3)	0,90	32
(63, 56, 4)	0,89	31
(63, 51, 5)	0,81	28
(63, 50, 6)	0,79	23
(63, 45, 7)	0,71	24
(63, 44, 8)	0,70	23
(63, 39, 9)	0,62	28
(127, 120, 3)	0,94	64
(127, 119, 4)	0,94	63
(127, 113, 5)	0,89	56
(127, 112, 6)	0,88	55
(127, 106, 7)	0,83	48
(127, 105, 8)	0,83	55
(127, 99, 9)	0,78	56

Tabelle 2: Gewichte des Kontrollpolynoms $h(x)$ und Koderaten für die untersuchten BCH-Kodes. Hervorgehoben sind die bei iterative *soft-decision*-Dekodierung leistungsfähigsten Kodes.

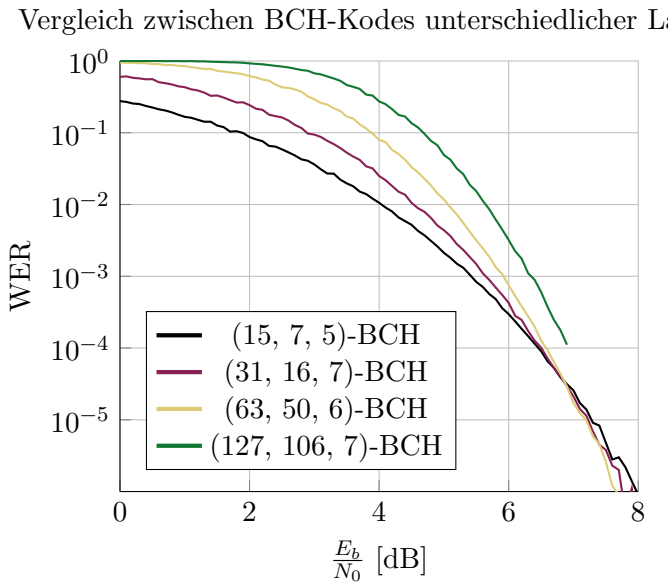

Abbildung 19: Vergleich der WER-Kurven von BCH-Kodes verschiendener Länge bei iterativer Dekodierung mittels *self-correcting Min-Sum* Variante 2.

2.6.2 BSC

Tabelle 3 listet den prozentualen Anteil nicht korrigierbarer oder falsch korrigierter Fehlermuster mit dem entsprechenden Gewicht $w(\mathbf{e})$ auf. Die garantierten Korrektureigenschaften des *hard-decision* Algorithmus PGZ (und ebenfalls Berlekamp-Massey) werden aufgezeigt. Die iterativen *soft-decision* Algorithmen können nur einen Teil der Fehler mit Gewicht $w(\mathbf{e}) \leq f_k$ korrigieren, dafür aber auch einen Teil der Fehler mit Gewicht $w(\mathbf{e}) > f_k$. Der Anteil dieser korrigierbaren Fehler mit Gewicht $w(\mathbf{e}) > f_k$ ist jedoch durchweg $< 5\,\%$.

2.7 Zusammenfassung

Ich habe den iterativen *soft-decision min-sum*-Algorithmus in verschiedenen Varianten mit dem *hard-decision* Dekodierungsalgorithmus PGZ und BM für BCH-Kode der Länge 15, 31, 63 und 127 mit geplanten Mindestabständen d_e zwischen 3 und 9 verglichen.

Der Vorteil von PGZ (und Berlekamp-Massey) liegt im garantierten Korrekturvermögen falls $w(e) \leq f_k$, sowie der Geschwindigkeit der Dekodierung insbesodere, oder gerade bei, Softwareimplementierung. Iterative *soft-decision* Algorithmen geben keine Garantien für die Dekodierung einer bestimmten Anzahl von Fehlern. Sie können jedoch einen Teil der

$w(\mathbf{e})$	MS	NMS	OMS	2D-NMS	SCMS (1)	SCMS (2)	PGZ	BM
0	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
1	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
2	29.7 %	8.2 %	9.2 %	8 %	29.2 %	2.4 %	0 %	0 %
3	79.2 %	67.7 %	69.5 %	67.8 %	76.1 %	55.1 %	0 %	0 %
4	96.3 %	98.4 %	98.2 %	98.4 %	96.7 %	95.3 %	100 %	100 %
5	99.5 %	100 %	100 %	100 %	99.8 %	99.9 %	100 %	100 %
6	99.7 %	100 %	100 %	100 %	99.8 %	100 %	100 %	100 %

Tabelle 3: Prozentualer Anteil an Fehlermustern mit Gewicht $w(\mathbf{e})$ bei denen eine Falschdekodierung oder Dekodierversagen auftrat.

Fehlermuster mit $w(\mathbf{e}) \leq f_k$ und ebenfalls einen Teil der Fehlermuster mit $w(\mathbf{e}) > f_k$ korrigieren. Leider ist der Prozentsatz für $w(\mathbf{e}) > f_k$ mit $< 5\,\%$ recht gering.

Das realistischere Bi-AWGN-Kanalmodell zeigt die Überlegenheit von iterativer *soft-decision*-Dekodierung für Kanäle ohne *multipath scattering* und mit freier erster Fresnelzone bei Abwesenheit von Interferenz.

Doch auch unter den verschiedenen iterativen Verfahren gibt es Unterschiede. *Self-correcting Min-Sum* nach [10] ist über den gesamten simulierten $\frac{E_b}{N_0}$-Bereich besser als alle anderen iterativen Verfahren oder liegt gleich auf. *Self-correcting Min-Sum* in der Variante 1 ist nicht zu Empfehlen. Es ist der schlechteste *soft-decision*-Dekodieralgorithmus und fällt für einzelne Kodes sogar hinter unkodierten Nachrichten zurück. Dazwischen liegen die restlichen Variantionen von *Min-Sum*, teilweise eng beeinander. Die Parametrisierung des suboptimalen *Min-Sum*-Algorithmus um für den Approximationsfehler zu korrigieren funktioniert nicht für den (31, 16, 7)-BCH-Kodes, die Kodelänge, oder der Einfluss ist zu gering um eine deutliche Verbesserung herbeizuführen oder eine Kombination von diesen Möglichkeiten. Hier muss noch untersucht werden, ob eine Optimierung für andere Kodes gegebenenfalls Verbesserungen zeigen.

Unter den betrachteten Kodes der Länge 31 zeigt sich der (31, 16, 7)-BCH-Kode am geeignetsten für iterative *soft-decision*-Dekodierung. Unter den betrachteten Kodes der Länge 63 ist dies der (63, 50, 6)-BCH-Kode. Die Unterschiede der Kodes in der Koderate (0,52 und 0,79) und im Mindestabstand lassen darauf schließen, dass der meiste Einfluss auf die Leistungsfähigkeit der iterativen Dekodieralgorithmen tatsächlich von der Kodestruktur und damit von $h(x)$ und der Kontrollmatrix H abhängt.

Es sei noch angemerkt, dass für WER $< 10^{-4}$ die Anzahl an ausgewerteten Kodewörtern nicht ausreicht um verlässliche, statistisch signifikante Aussagen zu treffen. Die Unsicherheit zeigt sich auch im „zappeligen“ Ende der WER-Kurven.

Nicht betrachtet wurde die algorithmische Komplexität der Verfahren. Darüber soll hier nur kurz anekdotisch berichtet werden. Die *hard-decision* Algorithmen PGZ und Berlekamp-Massey benötigten die wenigste Rechenzeit. Unter den iterativen *soft-decision* Algorithmen liegen der unmodifizierte *min-sum*-Algorithmus und *offset Min-Sum* in etwa gleich auf, dicht gefolgt von NMS und SCMS-2. Am langsamsten sind der *2D-normalized min-sum*-Algorithmus sowie *self-correcting Min-Sum* in der Variante 1. Bei Letzterem dürfte der hohe Rechenzeitbedarf aber durch die schlechte Korrekturleistung bedingt sein. Wenn kein Kodewort gefunden wird, muss der Algorithmus alle 50 Iterationen abarbeiten.

Literatur

[1] Jinghu Chen, A. Dholakia, E. Eleftheriou, M.P.C. Fossorier, and Xiao-Yu Hu. Reduced-complexity decoding of ldpc codes. Communications, IEEE Transactions on, 53(8):1288–1299, Aug 2005.

[2] M.P.C. Fossorier, M. Mihaljević, and H. Imai. Reduced complexity iterative decoding of low-density parity check codes based on belief propagation. Communications, IEEE Transactions on, 47(5):673–680, May 1999.

[3] R.G. Gallager. Low-density parity-check codes. Information Theory, IRE Transactions on, 8(1):21–28, January 1962.

[4] Daniel Gorenstein, W. Wesley Peterson, and Neal Zierler. Two-error correcting bose-chaudhuri codes are quasi-perfect. Information and Control, 3(3):291 – 294, 1960.

[5] Yongmin Jung, Yunho Jung, Seongjoo Lee, and Jaeseok Kim. New min-sum ldpc decoding algorithm using snr-considered adaptive scaling factors. to be published in ETRI Journal, 2014.

[6] Minghua Liu and Lijun Zhang. Iterative hybrid decoding algorithm for ldpc codes based on attenuation factor. Frontiers of Electrical and Electronic Engineering, 7(3):279–285, 2012.

[7] P. Poocharoen and M.E. Magaña. Different perspective and approach to implement adaptive normalised belief propagation-based decoding for low-density parity check codes. Communications, IET, 6(15):2314–2325, October 2012.

[8] Valentin Savin. Self-corrected min-sum decoding of LDPC codes. In Frank R. Kschischang and En-Hui Yang, editors, 2008 IEEE International Symposium on

Information Theory, ISIT 2008, Toronto, ON, Canada, July 6-11, 2008, pages 146–150. IEEE, 2008.

[9] Xiaofu Wu, Yue Song, Ming Jiang, and Chunming Zhao. Adaptive-normalized/offset min-sum algorithm. Communications Letters, IEEE, 14(7):667–669, July 2010.

[10] Hai yang LIU, Wen ze QU, Bin LIU, Jiang peng LI, Shi dong LUO, and Jie CHEN. Novel modified min-sum decoding algorithm for low-density parity-check codes. The Journal of China Universities of Posts and Telecommunications, 17(4):1 – 5, 2010.

[11] Juntan Zhang and Marc Fossorier. Improved min-sum decoding of ldpc codes using 2-dimensional normalization.