5.6 Decodierung des Walsh-Hadamard Codes

In Abschnitt 5.1 wurde der Walsch-Hadamard Code definiert und seine Parameter bestimmt. Der Walsh-Hadamard-Code WH_k ist ein binärer $[2^k-1, k, 2^{k-1}]$-Code. Es können also $\lfloor \frac{d-1}{2} \rfloor = \lfloor \frac{2^k-2}{4} \rfloor = \lfloor \frac{n}{4} \rfloor$ Fehler eindeutig korrigiert werden. Diese hohe Korrekturrate ist bei Kanälen mit sehr viel Rauschen sinnvoll. Bei der Mariner 9 Mission der NASA wurde 1971 ein (punktierter) Walsh-Hadamard-Code (ein $[32, 6, 16]$-Code) eingesetzt um Bilder vom Mars zu übertragen. Er hat eine ähnliche Datenrate wie der 5-fache Wiederholungscode, kann aber Fehler besser korrigieren.
Uns interessieren hier allerdings Codes von größerer Länge um ein neues Decodierprinzip zu erläutern. Wir untersuchen hier die lokale Decodierung, das bedeutet, es werden nur wenige Codewortstellen „angefragt“ und für die Decodierung eines einzelnen Bits nicht das ganze Codewort betrachtet. Dies wird durch einen randomisierte Decoder realisiert, bei dem mit einer geringen Wahrscheinlichkeit einzelne Stellen auch falsch decodiert werden können.
Die Generatormatrix G des WH-Codes enthält alle Vektoren $\neq 0$ der Länge k über $\mathbb{F}_2$.
Sei $c \in WH_k = \langle m, x \rangle_{x \in \mathbb{F}_2^k \setminus \{0\}}$ ein fehlerfrei übertragenes Codewort, so bezeichnet c_{e_i} mit $i = 1, \ldots, k$ also die Komponente von c, die durch Multiplikation mit e_i in G entsteht. Wegen $c_{e_i} = \langle m, e_i \rangle = m_i$, können die Komponenten m_i von m direkt gefunden werden (von der systematischen Codierung ist dieses Vorgehen bereits bekannt). Ist kein Fehler aufgetreten, so kann das Nachrichtenwort also direkt aus dem Codewort an den entsprechenden Stellen abgelesen werden: $m = (c_{e_1}, \ldots, c_{e_k})$. Mit dieser Vorüberlegung können wir das folgende Lemma einfach beweisen.

Lemma 25. *Es sei WH_k der Walsh-Hadamard-Code Länge $n = 2^k - 1$. Dann existiert ein randomisierter Decoder für WH_k, welcher bis zu $(\frac{1}{4} - \epsilon) \cdot n$ Fehler decodiert.*

Beweis. Sei $\tilde{c} = c + e$ das empfangene Wort, c ein Codewort und e der Fehlervektor vom Gewicht $\leq \left(\frac{1}{4} - \epsilon\right) \cdot n$. Es ist also höchstens ein $\left(\frac{1}{4} - \epsilon\right)$-Anteil von $\tilde{c}$ fehlerbehaftet.
Für jede (zufällig gewählte) Stelle ist die Fehlerwahrscheinlichkeit $\leq (\frac{1}{4} - \epsilon)$, also gilt für alle $x \in \mathbb{F}_2^k \setminus \{0\}$, dass $\langle m, x \rangle \neq \tilde{c}_x$ mit Wahrscheinlichkeit $\leq \left(\frac{1}{4} - \epsilon\right)$.
Sei nun $x \in \mathbb{F}_2^k$ zufällig gezogen, dann ist $wkt\{\tilde{c}_x \neq \langle m, x \rangle\} \leq \frac{1}{4} - \epsilon$ und ebenso auch für den „verschobenend Vektor“ $wkt\{\tilde{c}_{x+e_i} \neq \langle m, x + e_i \rangle\} \leq \frac{1}{4} - \epsilon$.
Fasst man dies beiden Ereignisse zusammen, so ergibt sich mit der folgenden Rechnung:

$$\begin{aligned} c_x + c_{x+e_i} &= \langle m, x \rangle + \langle m, x + e_i \rangle \\ &= \langle m, x \rangle + \langle m, x \rangle + \langle m, e_i \rangle \\ &= \langle m, e_i \rangle \\ &= m_i \end{aligned}$$

und $wkt\{\tilde{c}_x \neq c_x\} = wkt\{\tilde{c}_{x+e_i} \neq c_{x+e_i}\} \leq \frac{1}{4} - \epsilon$.
Bei einem beliebigen x ergibt sich das richtige m_i, falls beide Werte richtig empfangen wurden, also $\tilde{c}_x = c_x$ und $\tilde{c}_{x+e_i} = c_{x+e_i}$, oder falls beide Werte falsch empfangen wurden. Die Wahrscheinlichkeit dafür ist $\geq (3/4 + \epsilon)^2 + (1/4 - \epsilon)^2 = 5/8 + \epsilon/2 + \epsilon^2 =: 1/2 + \delta$.
Da die Erfolgswahrscheinlichkeit $> \frac{1}{2}$ kann durch l-faches Wiederholen und Mehrheitsentscheid die Fehlerwahrscheinlichkeit beliebig reduziert werden. Bei l-fachem Wiederholen ist die Fehlerwahrscheinlichkeit $\leq 2^{-\delta \cdot l}$ (Chernoff Schranke). □

Die Werte l und δ werden hier nicht näher betrachtet, einige offensichtlichen Randbedingungen sind: $l \leq n$, ansonsten ist l-faches Wiederholen nicht sinnvoll, und $l > 1/\delta$, ansonsten ist die Fehlerwahrscheinlichkeit nach dem Mehrheitsentscheid $\geq 1/2$.
Das Vorgehen im Beweis ergibt den folgenden

Decodieralgorithmus für das i-te Nachrichtenbit:

1. Wähle x zufällig,
2. berechne $\tilde{c}_x + \tilde{c}_{x+e_i}$ ($= m_i$ mit der Wahrscheinlichkeit $\geq \frac{1}{2} + \delta$).
3. Wiederhole 1. und 2. genügend oft;
4. ein Mehrheitsentscheid liefert dann m_i.

Innerhalb der Decodierkugeln ist klar was der Algorithmus liefern sollte. Bei einer größeren Fehlermenge kann er auch korrekt arbeiten; die Grenzen sind aber nicht einfach zu finden: kann l genÃŒgend gross werden, so könnten nach dem Beweis von oben bis zu $(1/2 - \epsilon) \cdot n$ Fehler korrigiert werden. Bei einer solchen Menge an Fehlern kann (wegen des Minimalabstands) aber ein anderes Codewort sehr nahe am empfangenen Wort liegen, so dass nicht klar ist was der Decodieralgorithmus liefern sollte. Bei dem Algorithmus wird jedes Nachrichtenbit einzeln, getrennt von den anderen, betrachtet; wird eines der Bits falsch decodiert, so hat das entsprechende Codewort einen Abstand $\approx n/2$ vom ursprÃŒnglichen Codewort.
Im Folgenden betrachten wir einen Algorithmus, der auch bei höheren Fehlerraten $> \frac{n}{4} \approx \frac{d-1}{2}$ noch sinnvolle Ergebnisse liefert.

5.7 Walsh-Hadamard-Code / List-Decodierung

Befindet sich ein empfangenes Wort innerhalb einer Decodierkugel, so können wir es (mit vorigen Algorithmus) decodieren. Wir stellt uns nun die Frage: „Wieviel (empfangene) Worte befinden sich innerhalb der Decodierkugeln?“
Das folgende Beispiel gibt ein ernüchterndes Bild.

Beispiel. Für den Walsh-Hadamard-Code WH_k der Länge n sind für ein paar kleine Werte von k der Radius der Decodierkugeln und der prozentuale Anteil der Kugeln im gesamten Raum $\mathbb{F}_2^n$ angegeben. Auch außerhalb der Kugeln gibt es viele Worte $\tilde{c}$ bei denen ein einzelnes Codewort existiert, das $\tilde{c}$ am nächsten liegt und damit eindeutig decodierbar ist. Ist man beim Decodieren auch mit einer kurzen Liste von Codeworten (und ihrem Abstand zu $\tilde{c}$) zufrieden, so spricht man von List-Decoding. Der Prozentsatz an Worten, bei denen eine Liste mit bis zu 3 Codeworten mit demselben Abstand zu $\tilde{c}$ ausreicht ist in der Tabelle ebenfalls angegeben.

k	n	Kugelradius	% Kugeln	eindeutig decodierbar	1-3 Codeworte gleich weit
4	15	3	28 %	48 %	80 %
5	31	7	5,3 %	52 %	85 %
6	63	15	0,1 %	58 %	90 %
7	127	31	0,00004 %	65 %	94 %

Diese Beispiel legt die Idee der List-Decodierung nahe. Zu einem empfangenen Wort wird eine Liste möglicher Codworte bzw. Nachrichtenworte ausgegeben die dem empfangenen Wort nahe sind.

Satz 26. ***(Goldreich-Levin)*** *Für jedes $\epsilon > 0$ existiert ein effizienter List-Decoder für WH_k, welcher $\left(\frac{1}{2} - \epsilon\right) \cdot n$ Fehler korrigiert und eine poly(k)-lange Liste an Kandidaten ausgibt.*

Diesen Satz werden wir nicht exakt beweisen. Wir geben hier den Algorithmus, der eine solche Liste liefert an und Begründungen weshalb er funktioniert:
Wir modifizieren den Algorithmus von vorher so, dass er auch noch bei mehr als $\frac{d-1}{2}$ Fehlern decodieren kann. In seiner ursprünglichen Form ist nicht klar, was der Algorithmus bei mehr als $\frac{d-1}{2}$ falschen Bits liefert und ist so nicht zu verwenden.
Sei nun an l Stellen $x_1, \ldots, x_l$ bekannt dass hier kein Fehler aufgetreten ist, also

$$\langle m, x_j \rangle = c_{x_j} = \tilde{c}_{x_j} \quad \text{für } j = 1, \ldots, l.$$

Für die anderen Stellen sei die Fehlerwahrscheinlichkeit $< \frac{1}{2} - \epsilon$. Dann kann m_i mit dem ursprünglichen Decoder für WH_k berechnet werden:

Modifizierter Decodieralgorithmus für das i-te Nachrichtenbit (Version 1):

1. Wähle x zufällig aus $\{x_1, \ldots, x_l\}$,
2. berechne $\tilde{c}_x + \tilde{c}_{x+e_i} = c_x + \tilde{c}_{x+e_i}$ $(= m_i$ mit der Wahrscheinlichkeit $\geq \frac{1}{2} + \epsilon)$.
3. Wiederhole 1. und 2. genügend oft, z.B. für alle $x_1, \ldots, x_l$;
4. ein Mehrheitsentscheid liefert dann m_i.

Bei l Wiederholungen ist die Fehlerwahrscheinlichkeit $\leq \dfrac{1}{2^{\epsilon \cdot l}}$
Dieser leicht modifizierte Decodieralgorithmus funktioniert auch bei einer höheren Fehlerrate (sofern l passend gewählt wird), die wichtige, zusätzliche Bedingung ist aber, dass wir l Stellen „sicher kennen“. Wie können wir dieses Problem angehen? Eine recht ungewöhnliche, aber sehr einfache Herangehensweise ist:

Idee. *Für die l Stellen im empfangenen Wort $c_{x_1}, \ldots, c_{x_l} \in \mathbb{F}_2$ gibt es (nur) 2^l viele Möglichkeiten. Eine davon ist die richtige.*
Probiere alle Möglichkeiten durch (die empfangenen Werte von $\tilde{c}_{x_i}$ werden nicht beachtet).

Modifizierter Decodieralgorithmus (Version 2):

1. *Berechne für jede Belegung von $\{c_{x_1}, \ldots, c_{x_l}\}$:*
   - *berechne $(m_1, \ldots, m_k)$ mit dem modifizierten Decodieralgorithmus (Version 1)*
   - *teste durch Codieren von $(m_1, \ldots, m_k)$ ob diese Nachricht möglich ist, d.h. die erwartete Fehler-Wahrscheinlichkeit hat.*
   - *speichere oder verwerfe $(m_1, \ldots, m_k)$.*

2. Gebe die gespeicherte Liste von Nachrichtenwörtern aus.

Die Decodierung (Version 1) liefert zwar immer noch einzelne Nachrichtenbits m_i unabhängig von den anderen Teilen der Nachricht. Anschließend wird aber die gesamte Nachricht durch Codieren auf Konsistenz überprüft.
Dieser Algorithmus liefert auf jeden Fall die gesuchte Liste von Nachrichtenworten, der Aufwand ist aber mit 2^l Durchläufen des „Modifizierten Decodieralgorithmus (Version 1)“ zu groß.
Da es sich bei der Codierung um eine lineare Abbildung handelt lassen sich l richtige Stellen $c_{x_1}, \ldots, c_{x_l} \in \mathbb{F}_2$ aber auch viel schneller finden:

- Setze $t := \lceil \log(l) \rceil$.
- Wähle $s_1, \ldots, s_t \in \mathbb{F}_2^k$ (wie x_j vorher).
- Sind $\tilde{c}_{s_j} = c_{s_j}$ für $j = 1, \ldots, t$ fehlerfrei übertragen, dann ist auch $c_{s_1} \oplus c_{s_2} = \langle m, s_1 \rangle \oplus \langle m, s_2 \rangle = \langle m, s_1 \oplus s_2 \rangle$ fehlerfrei und damit sind alle 2^t vielen Stellen $\sum b_j c_{s_j} = c_{\sum b_j s_j}$ für $b_1, \ldots, b_t \in \mathbb{F}_2$ fehlerfrei.

Durch dieses Vorgehen erzeugen wir einen korrekten Satz von l Stellen für das empfangene Wort indem wir $O(2^t) = O(l)$ viele Belegungen durchprobieren. Dies liefert einen effizienten List-Decoder für den Walsh-Hadamard-Code, der bis zu $\left(\frac{1}{2} - \epsilon\right) \cdot n$ Fehler „korrigieren“ kann.