Mathematisches Kolloquium
Universität Bayreuth

UNIVERSITÄT BAYREUTH

Konstruktion von guten linearen Blockcodes

Markus Grassl

in Zusammenarbeit mit Greg White, University of Sydney

Institut für Algorithmen und Kognitive Systeme
Fakultät für Informatik
Universität Karlsruhe (TH)
http://iaks-www.ira.uka.de/home/grassl

Übersicht

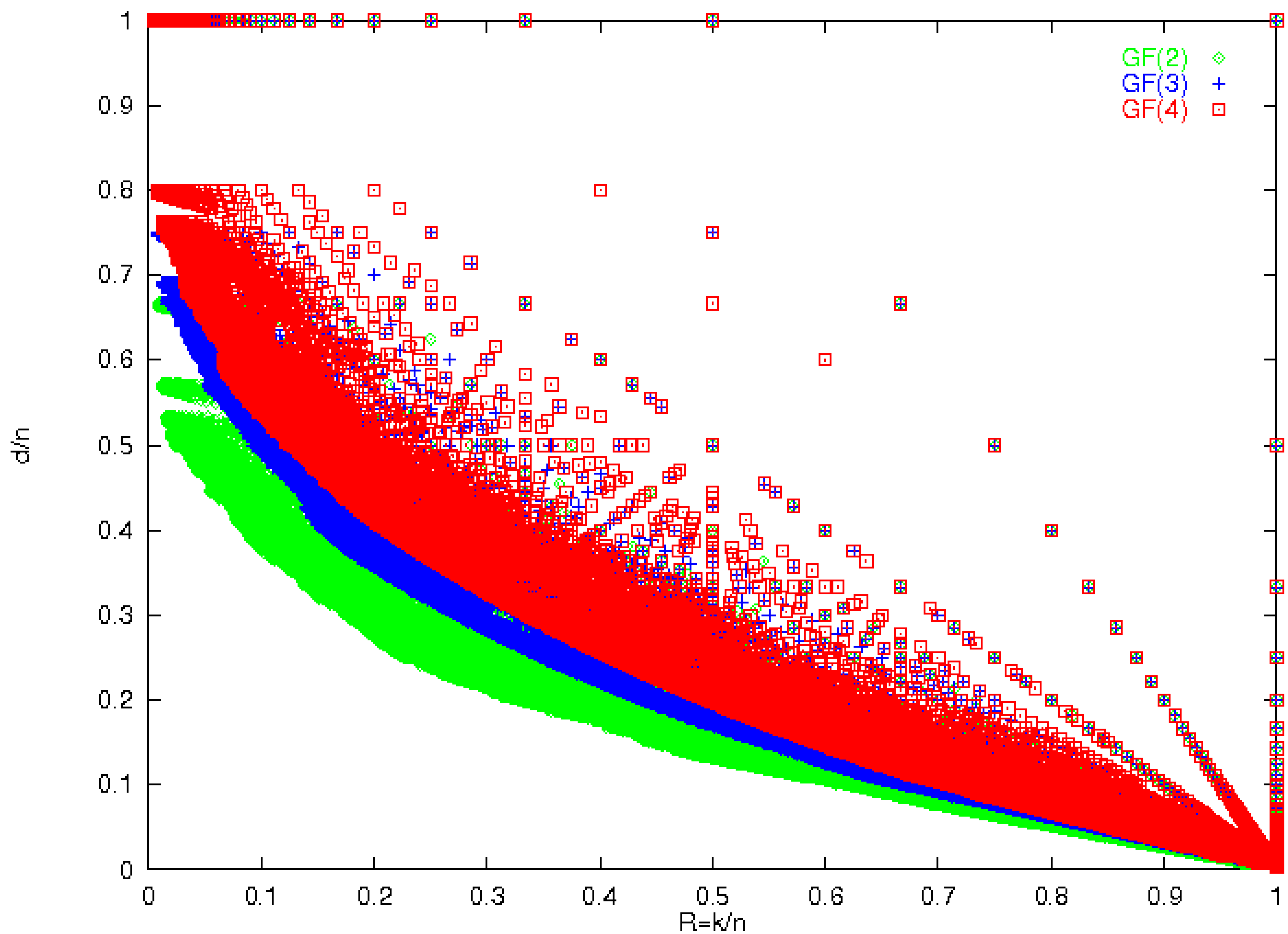
- lineare Blockcodes
- Brouwers Tabellen
- Das BKLC-Projekt
- Berechnung der Minimaldistanz
- Punktierung an bestimmten Stellen
- Kombination von Codes

Lineare Blockcodes

$C = [n, k, d]_q$

- endlicher Körper $\mathbb{F}_q$ als Alphabet
- Blockcode der Länge n
- Untervektorraum der Dimension k von $\mathbb{F}_q^n$
  - Zeilenraum der Generatormatrix $G \in \mathbb{F}_q^{k \times n}$
  - Kern der Kontrollmatrix $H \in \mathbb{F}_q^{(n-k) \times n}$
- Minimaldistanz d
  - Erkennung von bis zu $d - 1$ Fehlern
  - Korrektur von $t \leq \left\lfloor \frac{d-1}{2} \right\rfloor$ Fehlern
- Rate $R = \frac{k}{n}$

Minimaldistanz/Rate



Brouwers Tabellen

Ziel: Finde gute Codes, d. h., d/n möglichst groß bei gegebener Rate k/n.

Brouwers Tabellen untere/obere Schranken für die Minimaldistanz d von linearen Blockcodes $C = [n, k, d]_q$ über $\mathbb{F}_q$ für $q = 2, 3, 4, 5, 7, 8, 9$

aktuelle Version unter
`http://www.win.tue.nl/~aeb/voorlincod.html`

Problem: wenig Information zur *Konstruktion* der Codes

Projekt:

- explizite Konstruktion von Codes, die die untere Schranken erreichen
- Verbesserung der unteren Schranken

Das Projekt *Best Known Linear Codes* (BKLC)

Ziel: „Datenbank“ mit *Konstruktionen* für die Codes

- Kooperation mit der *Computational Algebra Group, University of Sydney* (Entwickler von Magma): John Cannon, Damien Fisher, Greg White
- Implementierung vieler Konstruktionsverfahren
- Algorithmen zur Verifikation der Codes
- zeitlicher Verlauf
  - Juni 1996: E-Mail-Diskussion über Codierungstheorie & Magma
  - Ende 1999: Beginn des BKLC-Projekts
  - Juli 2001 (V2.8): Codes über $\mathbb{F}_2$ für $n \leq 256$
  - April 2003 (V2.10): Codes über $\mathbb{F}_4$ für $n \leq 100$
  - Mai 2004 (V2.11): Codes über $\mathbb{F}_3$ für $n \leq 100$

BKLC-Statistik

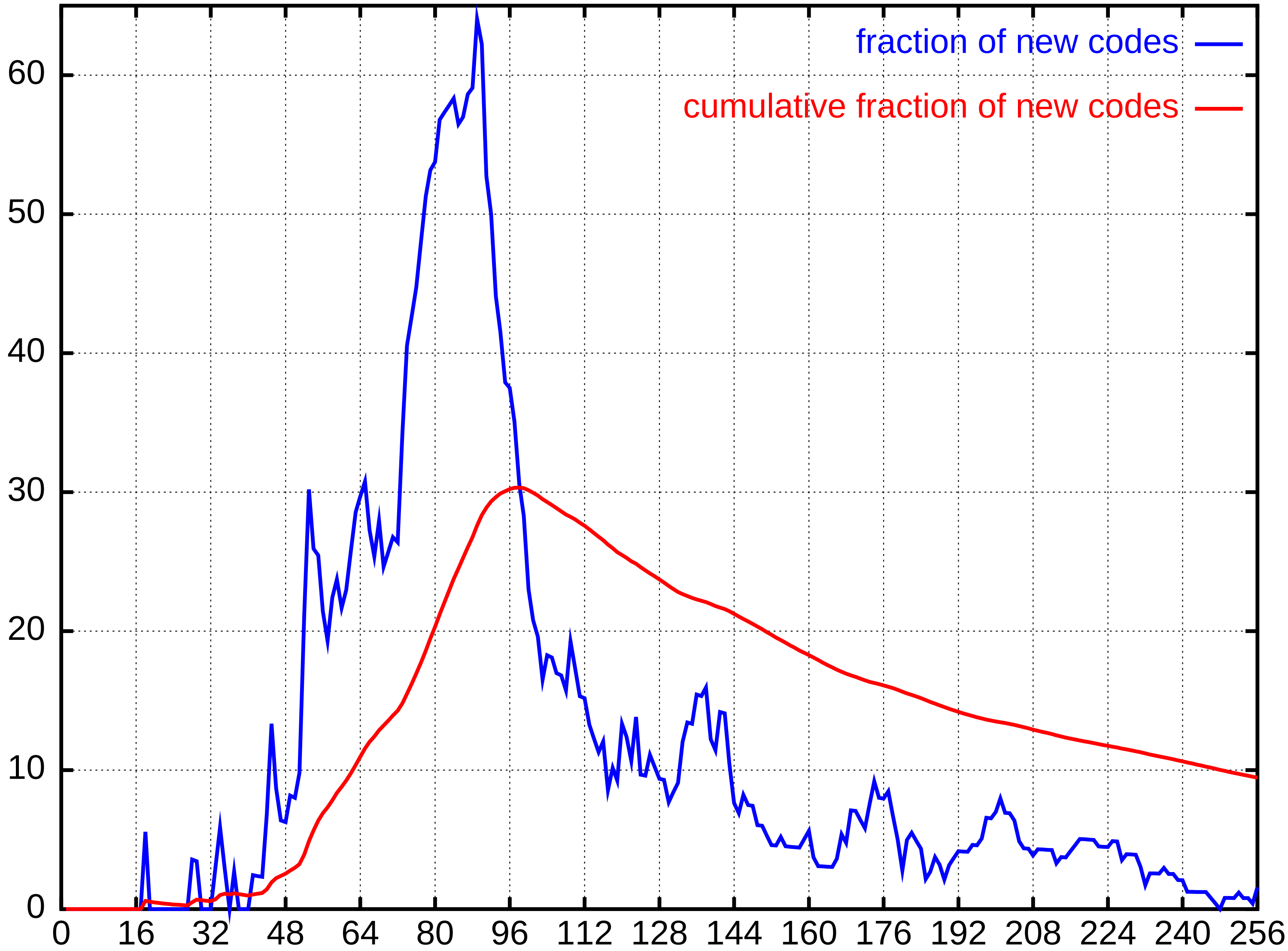
Verbesserungen

q	2	3	4	5	7	8	9
verbesserte Codes	1700	3417	3320	1279	87	662	509

fehlende Codes

q	$n_{\max}$	SEED	gesamt	SEED_{100}	gesamt_{100}
2	256	–	–	–	–
3	243	1211	7302	–	–
4	256	1746	12213	6	13
5	130	144	66	50	283
7	50	–	–	137	560
8	130	399	1988	132	616
9	130	294	1421	33	87

Stand: Dezember 2005



Verbesserung der unteren Schranken von Codes über $\mathbb{F}_4$

Stand: 03.02.2004

Code-Konstruktionen (I)

Direkte Konstruktionen

- zyklische Codes
- AG-Codes
  (Kurven mit vielen rationalen Punkten)
- Codes von kombinatorischen Objekten

Sekundäre Konstruktionen

- Verkürzen, Punktieren, Verlängern
- Verkürzen und Punktieren an bestimmten Positionen
- Kombination von Codes

$\Longrightarrow$ untere Schranken für die Minimaldistanz

Berechnung der Minimaldistanz (I)

- theoretische untere Schranken
  - „konstruktiv“: z. B. BCH-Schranke, AG-Codes
  - „Abzählung“: z. B. Gilbert-Varshamov-Schranke
- Die Berechnung der Minimaldistanz ist NP-hart [Vardy 97].
- naive Methoden:
  - Aufzählen aller Codeworte

$$d = \min\left\{\operatorname{wgt}(\boldsymbol{c}) : \boldsymbol{c} \in C \setminus \{\boldsymbol{0}\}\right\}$$

  - Aufzählen aller Worte kleinen Gewichts, bis ein Codewort gefunden wurde

$$d = \min\left\{w : w = 1, \ldots, n \mid \exists \boldsymbol{v} \in \mathbb{F}_q^n : \boldsymbol{v} \in C\right\}$$

Berechnung der Minimaldistanz (II)

Systematische Codierung

Codierung von $\boldsymbol{i} \in \mathbb{F}_q^k$ mit einer systematischen Generatormatrix $G = (I|A)$
$\Longrightarrow \boldsymbol{c} = \boldsymbol{i}G = (\boldsymbol{i}, \boldsymbol{i}A)$ mit $\mathrm{wgt}(\boldsymbol{c}) \geq \mathrm{wgt}(\boldsymbol{i})$

Aufzählen und Codieren aller Worte $\boldsymbol{i} \in \mathbb{F}_q^k$ vom Gewicht $\mathrm{wgt}(\boldsymbol{i}) \leq w$, d. h.

$$S := \{\boldsymbol{i}G : \boldsymbol{i} \in \mathbb{F}_q^k | \mathrm{wgt}(\boldsymbol{i}) \leq w\}$$

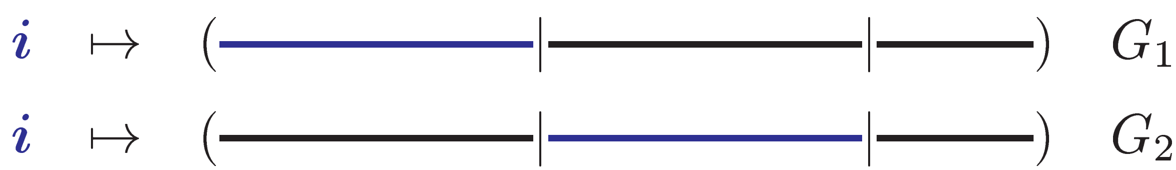
$$\begin{array}{llll} \Longrightarrow \quad d & \leq & \min\big\{\mathrm{wgt}(\boldsymbol{c}) : \boldsymbol{c} \in S \setminus \{\boldsymbol{0}\}\big\} & \text{obere Schranke} \\ & & \min\big\{\mathrm{wgt}(\boldsymbol{c}) : \boldsymbol{c} \in C \setminus S\big\} \quad \geq \quad w+1 & \text{untere Schranke} \end{array}$$

untere Schranke für das Gewicht der noch nicht aufgezählten Codeworte

Berechnung der Minimaldistanz (III)

Mehrere Generatormatrizen

systematische Generatormatrizen mit disjunkten Informationsmengen



t Generatormatrizen $\Longrightarrow$ untere Schranke $t \cdot (w+1)$ [Brouwer]

Verbesserungen [Zimmermann], [Grassl & White]

- überlappende Informationsmengen
- Größe der Schnittmenge vermindert den Beitrag zur unteren Schranke
- Vorberechnung, welche Matrizen zur unteren Schranken beitragen

Berechnung der Minimaldistanz (IV)

Aufwand

- Anzahl M der aufgezählten Codeworte

$$M = t \sum_{w=1}^{w_{\max}} \binom{k}{w} (q-1)^{w-1}$$

liefert untere Schranke $d_{\min} \geq t \cdot (w_{\max} + 1)$
(bei t disjunkten Informationsmengen)

- normierte Vektoren
- *Revolving-door*-Algorithmus für die w-Teilmengen von $\{1, \ldots, k\}$
- verallgemeinerter Gray-Code für die Vektoren

$\Longrightarrow$ nur eine Vektoroperation pro Codewort (plus Vorberechnung)

Zyklische Codes

Zyklisch verschobene Informationsmengen

[Chen 70, Coppersmith & Seroussi 84, Kschischang & Pasupathy 92]

Länge $n = 18$

Dimension $k = 7$

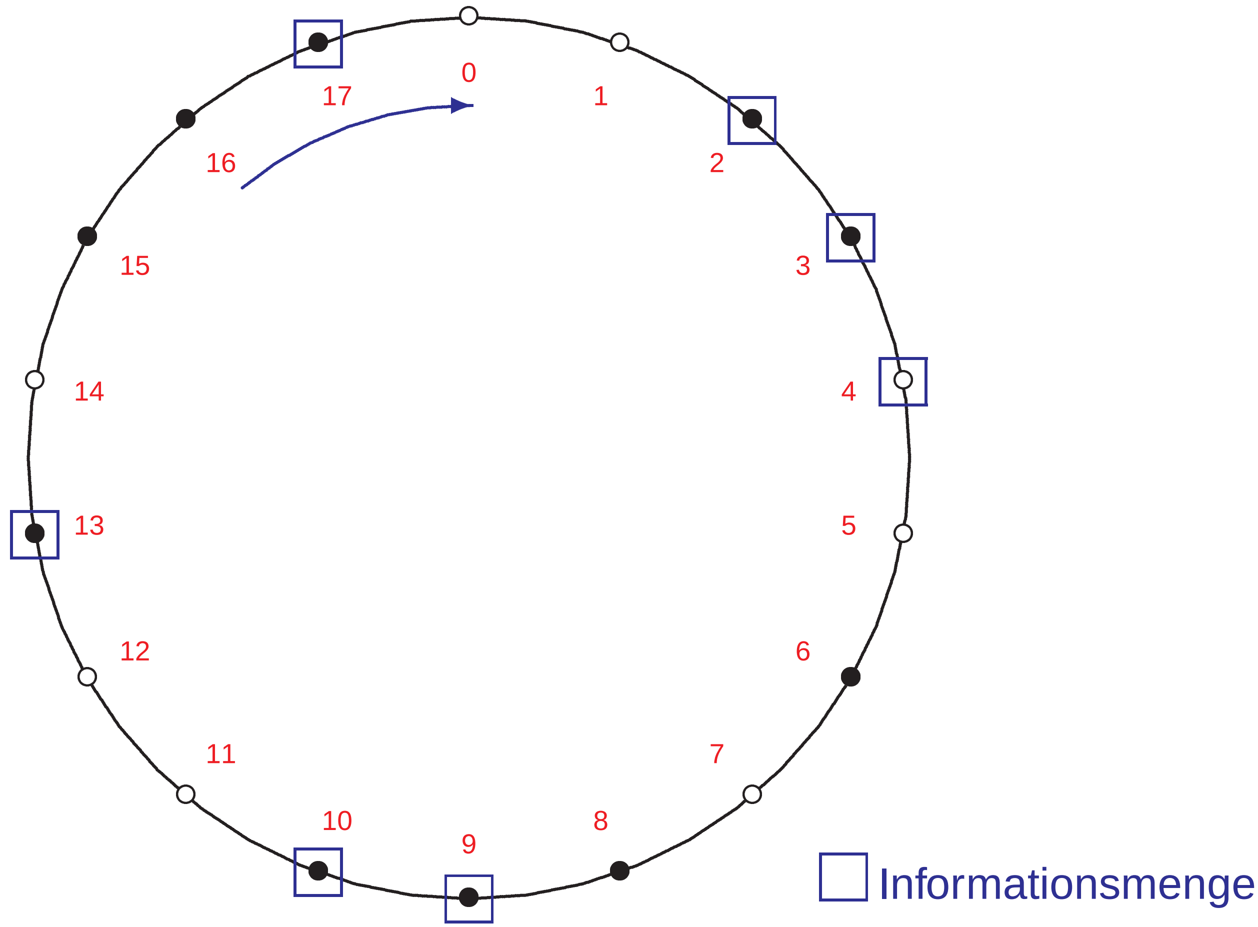
Gewicht $w = 10$

Informationsgewicht $w_{\mathcal{I}} = 6$

mittleres Gewicht:

$\overline{w_{\mathcal{I}}} = \mathrm{wgt}(\boldsymbol{c})k/n$

hier: $35/9 \approx 3.89$



Zyklische Codes

Zyklisch verschobene Informationsmengen

[Chen 70, Coppersmith & Seroussi 84, Kschischang & Pasupathy 92]

Länge $n = 18$

Dimension $k = 7$

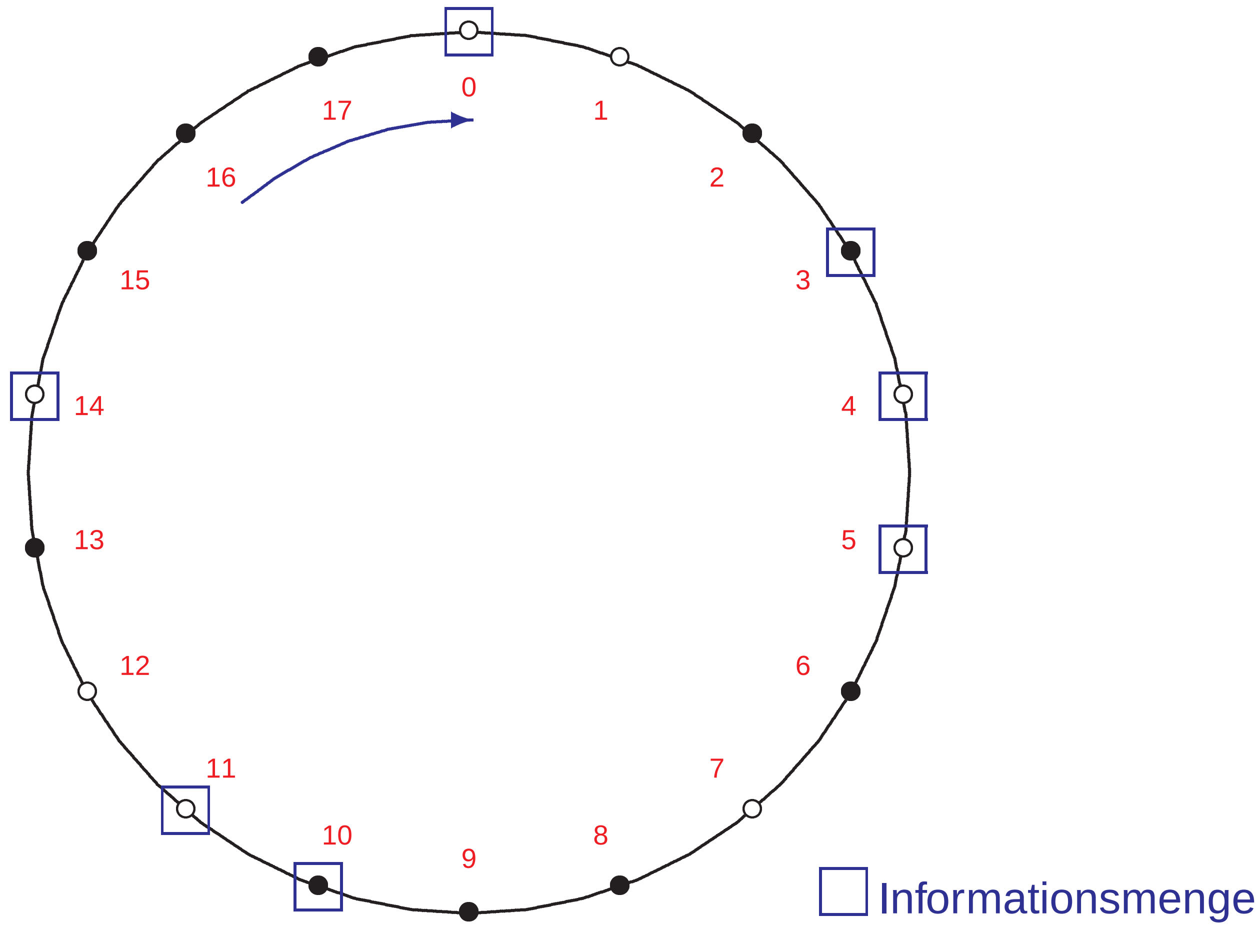
Gewicht $w = 10$

Informationsgewicht $w_{\mathcal{I}} = 2$

mittleres Gewicht:

$\overline{w_{\mathcal{I}}} = \mathrm{wgt}(\boldsymbol{c})k/n$

hier: $35/9 \approx 3.89$



Zyklische Codes

Zyklisch verschobene Informationsmengen

[Chen 70, Coppersmith & Seroussi 84, Kschischang & Pasupathy 92]

Länge $n = 18$

Dimension $k = 7$

Gewicht $w = 10$

Informationsgewicht $w_{\mathcal{I}} = 2$

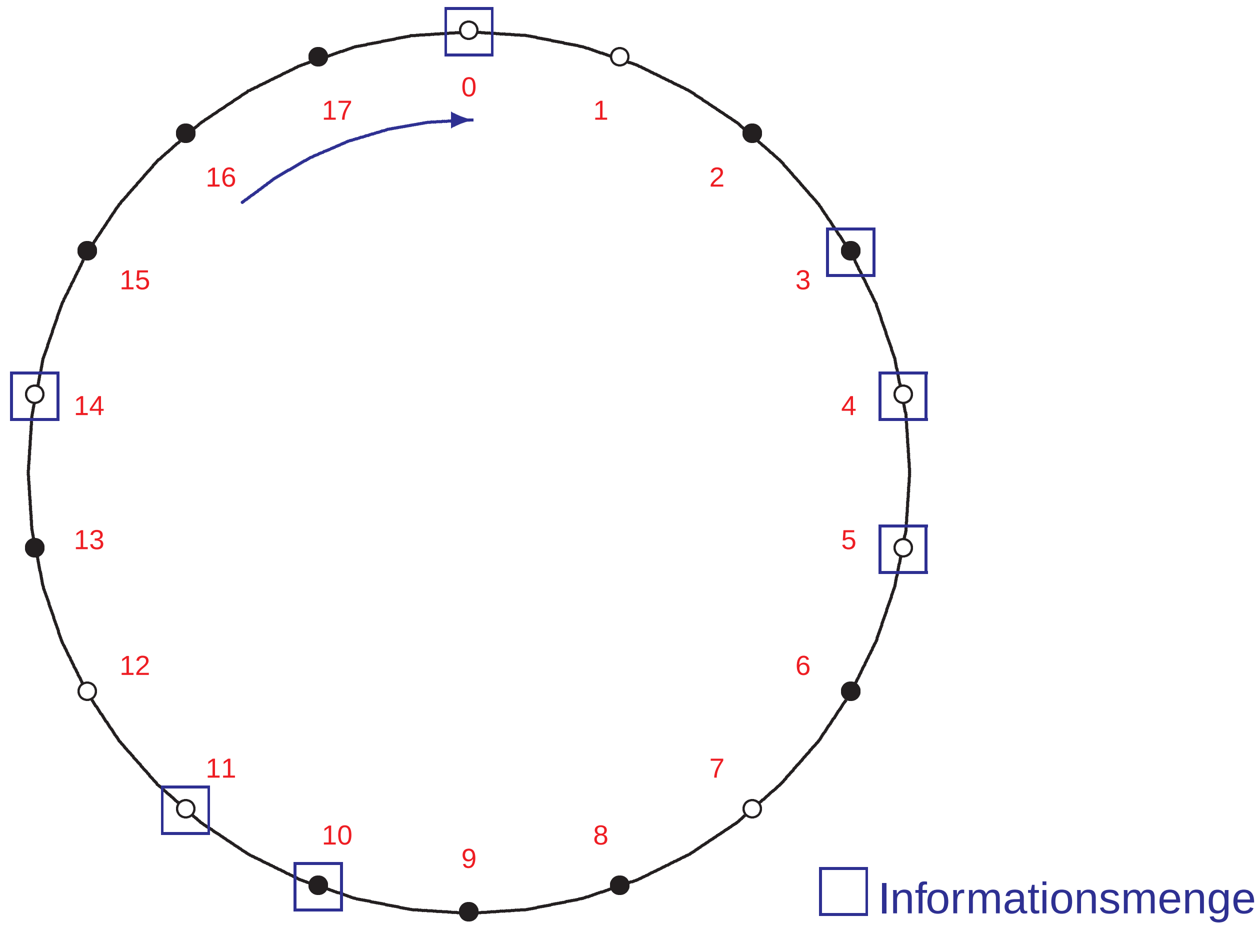
mittleres Gewicht:

$\overline{w_{\mathcal{I}}} = \mathrm{wgt}(\boldsymbol{c})k/n$

hier: $35/9 \approx 3.89$

untere Schranke:

$\mathrm{wgt}(\boldsymbol{c}) \geq \lceil (w+1)n/k \rceil$



Quasi-zyklische Codes

- linearer Blockcode der Länge $n = mt$
- t Blöcke der Länge m
- invariant bzgl. simultaner zyklischer Verschiebung

$$\left(c_1 c_2 \ \ldots\ c_m \;\middle|\; c_{m+1} \ldots c_{2m} \;\middle|\; \cdots \;\middle|\; c_{n-m+1} \ldots c_n \right)$$

- max. Rang in jedem Block:
  - „zyklische“ untere Schranke für jeden Block $\lceil (w+1)m/k \rceil = \lceil (w+1)n/(kt) \rceil$
  - untere Schranke: $t\lceil (w+1)n/(kt) \rceil$
- Erweiterungen [G. White, submitted]
  - Rang $< k$
  - beliebige zykl. Verschiebungen aus der Automorphismengruppe

Standard-Punktierung

Punktiere einen Code an m Positionen durch Streichen von m Koordinaten

$$[n, k, d] \Longrightarrow [n-m, k, d-m]$$

- Sind nur die Parameter $C = [n, k, d]$ bekannt, so ist keine Verbesserung möglich.
- Ist der Code explizit gegeben, kann man z. B. die Menge aller Codeworte minimalen Gewichts verwenden, um geeignete m Positionen zu finden.

Punktierung: Vorüberlegungen (I)

- $C = [n, k, d]$ habe genau ein minimales Codewort:

$$\boldsymbol{c} = 00110\ldots1010$$

Streichen der Position i liefert:

$$\begin{array}{ccccc} i \in \overline{\mathrm{supp}}(\boldsymbol{c}) & \Longrightarrow & c_i = 0 & \Longrightarrow & [n-1, k, d] \\ i \notin \overline{\mathrm{supp}}(\boldsymbol{c}) & \Longrightarrow & c_i \neq 0 & \Longrightarrow & [n-1, k, d-1] \end{array}$$

Punktierung: Vorüberlegungen (II)

- zwei minimale Codeworte $\boldsymbol{c_1}$ und $\boldsymbol{c_2}$:

$$\begin{aligned} \boldsymbol{c_1} &= 00110\ldots1010 \\ \boldsymbol{c_2} &= 10011\ldots0011 \end{aligned}$$

$$\left.\begin{array}{rr} & i \in \overline{\mathrm{supp}}(\boldsymbol{c_1}) \\ \text{und} & i \in \overline{\mathrm{supp}}(\boldsymbol{c_2}) \end{array}\right\} \Longrightarrow [n-1,k,d]$$

$$\text{sonst} \Longrightarrow [n-1,k,d-1]$$

- allgemein:
  Streiche m Positionen P, so daß in jedem minimalen Codewort mindestens eine Null-Position gestrichen wird.
  $\Longrightarrow [n-m,k,d-m+1]$

Punktierung: *Hitting Set*

Definition:

Sei $\mathcal{S}$ eine Menge von Teilmengen von $\{1, \ldots, n\}$.

Eine Teilmenge $I \subseteq \{1, \ldots, n\}$ heißt *hitting set* von $\mathcal{S}$, falls I mindestens ein Element von jeder Teilmenge in $\mathcal{S}$ enthält, d. h. $\forall S \in \mathcal{S}\colon\ S \cap I \neq \emptyset$.

Theorem:

Sei W_C die Menge aller Worte minimalen Gewichts in $C = [n, k, d]$.

Falls es ein *hitting set* von $\{\overline{\mathrm{supp}}(\boldsymbol{c}) : \boldsymbol{c} \in W_C\}$ der Größe m gibt, dann existiert ein Code $C' = [n - m, k, d - m + 1]$.

Spezielle Punktierung

- Berechne die Menge W_C aller Codeworte minimalen Gewichts: Adaption des Algorithmus zur Berechnung der Minimaldistanz.

  explizite Berechnung der Minimaldistanz möglich
  $\Longrightarrow W_C$ kann berechnet werden

- Berechne ein *hitting set* mit möglichst wenigen Elementen.
- Suche Codes $C = [n, k, d]$ in der Datenbank mit
  - explizite Berechnung von W_C ist möglich
  - für ein möglichst großes m gilt für die untere Schranke d_{lb}

$$d_{\text{lb}}(n-m,k) = d_{\text{lb}}(n,k) - m$$

(m ist eine obere Schranke für die Größe eines guten *hitting set*)

Berechnung eines guten *Hitting Set*

- NP-hart
- Greedy-Algorithmus liefert relativ gute Ergebnisse
- lineare Optimierung (über $\mathbb{R}$) liefert eine untere Schranke für $|I|$

$$\text{minimiere} \quad f(x) = \sum_{i=1}^{n} x_i$$

$$\text{mit } \forall S \in \mathcal{S}: \sum_{i \in S} x_i \geq 1 \text{ und } x_i \in \{0,1\}$$

Verallgemeinerung

Punktierung eines Codes $C = [n, k, d]$:

- Verbesserung der Minimaldistanz um 1:
  betrachte Codeworte vom Gewicht d
- Verbesserung der Minimaldistanz um 2:
  betrachte Codeworte vom Gewicht d und $d + 1$

$$\vdots$$

- Verbesserung der Minimaldistanz um r:
  betrachte Codeworte vom Gewicht d, $d + 1, \ldots, d + r - 1$

Verallgemeinertes *Hitting Set*

Seien $C = [n, k, d]_q$ ein linearer Code und

$$\mathcal{S}(w) := \{\overline{\mathrm{supp}}(\boldsymbol{c})\colon \boldsymbol{c} \in C \mid \mathrm{wgt}(\boldsymbol{c}) = w\}.$$

Eine Teilmenge $I \subseteq \{1, \ldots, n\}$ heißt *hitting set vom Grad r für C*, falls gilt

$$\begin{aligned} \forall S \in \mathcal{S}(d)\colon &\quad |S \cap I| \geq \quad r \\ \forall S \in \mathcal{S}(d+1)\colon &\quad |S \cap I| \geq \quad r-1 \\ &\quad\vdots \\ \forall S \in \mathcal{S}(d+r-1)\colon &\quad |S \cap I| \geq \quad 1 \end{aligned}$$

Verallgemeinertes *Hitting Set* & Punktierung

Theorem:

Sei $C = [n, k, d]_q$ ein linearer Code mit Minimaldistanz d.

Falls $I \subseteq \{1, \ldots, n\}$ ein *hitting set* vom Grad r der Größe $|I| = m$ ist, dann existitert ein Code $C' = [n - m, k, d - m + r]$.

Vereinfachungen:

- wiederholte Punktierung mit Verbesserung der Minimaldistanz um eins
- randomisierte Optimierung:
  Wähle eine Position i, die z. B. 80% der maximal möglichen Teilmengen überdeckt.

Code-Konstruktionen (II)

Construction X

Kombination zweier ineinander enthaltener Codes C_1 und C_2 mit C_3

gegeben: $C_1 = [n, k_1, d_1]_q$ und $C_2 = [n, k_2, d_2]_q$ mit $C_2 \subset C_1$, $k_2 < k_1$,
$C_3 = [n', k_1 - k_2, d_3]_q$

Ergebnis: $C = [n + n', k_1, d]_q$ mit $d \geq \min\{d_2, d_1 + d_3\}$
Generatormatrix

$$G = \begin{pmatrix} G_{12} & G_3 \\ G_2 & 0 \end{pmatrix}$$

Variationen

Construction XX, Construction X3, etc. kombinieren drei und mehr Codes

$\Longrightarrow$ finde gute Ketten von Codes
Kandidaten: zyklische/quasi-zyklische Codes, AG-Codes

Verbände von quasi-zyklischen Codes

Quasi-zyklischer Code mit einem Erzeuger

erzeugt von einem Zeilenvektor und seinen quasi-zyklischen Verschiebungen

$$\boldsymbol{g} = \big(g_1(X), g_2(X), \ldots, g_t(X)\big)$$

oder $$\boldsymbol{g} = \big(f_1(X)g_0(X), f_2(X)g_0(X), \ldots, f_t(X)g_0(X)\big)$$

falls alle zyklischen Codes gleich sind, gilt $g_0(X) | (X^m - 1)$

Notation: $C\langle g_0(X); f_1(X), \ldots, f_t(X)\rangle$

Lemma:

1. $a(X) \mid g_0(X)$
   $\Longrightarrow C \subseteq C\langle a(X); f_1(X), \ldots, f_t(X)\rangle$.

2. $g_0(X) \mid b(X) \mid (X^m - 1)$
   $\Longrightarrow C\langle b(X); f_1(X), \ldots, f_t(X)\rangle \subseteq C$.

$$\begin{array}{c} X^m - 1 \\ | \\ b(X) \\ | \\ g_0(X) \\ | \\ a(X) \end{array}$$

Beispiel

- ternärer quasi-zyklischer Code $C_1 = [195, 36, 75]_3$ mit $t = 5$
  Verifikation von $d_{\min} = 75$:
  - ca. 22 Tage mit einem AMD Opteron 250 (2,4 GHz), Magma V2.12
  - $35\,616\,895\,103\,240 \approx 2^{44}$ Codeworte aufgezählte ($\approx 2^{24}/s$)
  - ca. 9-mal schneller als mit 5 allgemeinen Informationsmengen
- quasi-zyklischer Teilcode $C_2 = [195, 32, 79]_3$
- neue Codes:
  $C = [199, 36, 77]_3$ mit $C_3 = [4, 3, 2]$ ($d_{\text{Brouwer}} = 75$)
  $C = [202, 36, 79]_3$ mit $C_3 = [7, 3, 4]$ ($d_{\text{Brouwer}} = 77$)

weitere Beispiele siehe [Grassl & White, Proceedings ISIT 05]

Zusammenfassung/Ausblick

- verbesserter Minimaldistanz-Algorithmus nutzt die Struktur von quasi-zyklischen Codes (und Varianten)
- Punktieren an bestimmten Positionen
- Anwendung von Construction X auf quasi-zyklische Codes (und andere Ketten von Codes)
- Suche/Konstruktion von Verbänden guter Codes

$\Longrightarrow$ viele Codes mit verbesserter Minimaldistanz

fehlende Codes

- untere Schranken via Abzählung
- Kurven mit vielen rationalen Punkten, explizite Gleichungen unbekannt