Studia i Materiały Informatyki Stosowanej, Tom 6, Nr 17, 2014
str. 10-15

OCENA JAKOŚCIOWA KODÓW LINIOWYCH O SKOŃCZONYCH CHARAKTERYSTYKACH

Mariusz Frydrych, Wojciech Horzelski, Dariusz Doliwa

Uniwersytet Łódzki
Wydział Matematyki i Informatyki
ul.Banacha 22, 90-238 Łódź
e-mail: frydrych@math.uni.lodz.pl
horzel@math.uni.lodz.pl
doliwa@math.uni.lodz.pl

Streszczenie: *Praca opisuje właściwości jakościowe kodów liniowych w wymiarze „połówkowym”, tzn. gdy wymiar kodu jest równy jego kowymiarowi. Przedstawiona została analiza odległości Hamminga dla kodów reprezentowanych w przestrzeni wektorowej nad ciałem skończonym charakterystyki większej niż dwa. Konstrukcja kodów została oparta o automorfizm Frobeniusa. Rozpatrywano kody o wymiarze trzy zanurzone w sześcioelementowej przestrzeni nad ciałem siedmioelementowym.*

Słowa kluczowe: *kody liniowe, kodowanie, odległość Hamminga*

Quality characteristics of finite linear codes

Abstarct: *Paper describes the qualitative properties of linear codes in the dimension of "half-width", ie. when the code dimension is equal to its co-dimension. The codes are constructed with use of Frobenius automorphism in a vector space over a finite field of the characteristic more than two. We considered three dimensions codes embedded in six dimensional space over the field with seven elements. An analysis of the Hamming distance for such a codes is also represented in the paper.*

Keywords: *Linear codes, coding, Hamming distance*

1. WSTĘP

Kody liniowe stosowane są powszechnie w przesyłaniu danych w zaszumionym medium transmisyjnym. Przez wymiar kodu rozumie się tu wielkość pojedynczego słowa kodowego, z kolei kowymiar kodu, mierzy tak zwaną nadmiarowość czyli ilość informacji niezbędnej do wykrywania i ewentualnej korekcji błędów przesyłanych danych, co łącznie daje przepustowość łącza. Liniowość kodu znakomicie upraszcza procesy kodowania i dekodowania, co skutkuje duża wydajnością implementowanych algorytmów. Z oczywistych powodów, najczęściej stosuje się kody binarne, co znacznie zawęża spektrum możliwej do uzyskania jakości kodu. Zastosowanie większej liczby stanów (ciał skończonych charakterystyki większej niż dwa), daje elastyczną strukturę kolekcji kodów liniowych. Wykorzystywana tutaj metoda szybkiego generowania takich kodów została przedstawiona w pracy [2].

2. KOD LINIOWY

Niech $k, n, p, w, q \in \mathbb{N}$, p – liczba pierwsza, $q = p^w$, $k \leq n$.

Definicja. *Każdą k-wymiarową podprzestrzeń wektorową C przestrzeni n-wymiarowej $\mathbb{F}_q^n$ nazywamy kodem liniowym o długości n, wymiaru k, nad ciałem $\mathbb{F}_q$* [1][4] .
Wybór bazy

$$B = (b_1, \dots, b_k),\ b_1, \dots, b_k \in C \subset \mathbb{F}_q^n$$

indukuje monomorfizm przestrzeni liniowych

$$\iota: \mathbb{F}_q^k \to \mathbb{F}_q^n, \begin{pmatrix} \xi_1 \\ \vdots \\ \xi_k \end{pmatrix} \mapsto \sum_{j=1}^{k} \xi_j b_j, \qquad im(\iota) = C$$

zwany *kodem liniowym.*
Dostajemy

$$0 \to \mathbb{F}_q^k \xrightarrow{\iota} \mathbb{F}_q^n \xrightarrow{\pi} \mathbb{F}_q^n / C \to 0$$

tzw. krótki ciąg dokładny przestrzeni wektorowych.

$$codim\ C = \dim \mathbb{F}_q^n / C = n - k.$$

Składając π z dowolnym izomorfizmem $\mathbb{F}_q^n / C \xrightarrow{\sim} \mathbb{F}_q^{n-k}$ dostajemy ponownie krótki ciąg dokładny.
Operator (macierz) *H* nazywamy *anihilatorem, macierzą kontrolną (check matrix)* kodu *C*.

$$0 \to \mathbb{F}_q^k \xrightarrow{\iota} \mathbb{F}_q^n \xrightarrow{H} \mathbb{F}_q^{n-k} \to 0$$

Kowymiar podprzestrzeni *codimC = n - k* to ilość stopni kontrolnych kodu (inaczej nadmiarowość), a wymiar *dimC = k* odpowiada zawartości informacji.
Wektory bazowe

$$(b_1, \ldots, b_k), b_1, \ldots, b_k \in \mathbb{F}_q^n$$

są liniowo niezależne, więc znajdziemy podciąg

$$1 \le j_1 < \cdots < j_k \le n,$$

taki że macierz

$$b_{j_1,\ldots,j_k} = \begin{bmatrix} b_{j_1,1} & \cdots & b_{j_1,k} \\ \vdots & \ddots & \vdots \\ b_{j_k,1} & \cdots & b_{j_k,k} \end{bmatrix}$$

jest nieosobliwa.

$$P \cdot B \cdot b_{j_1,\ldots,j_k}^{-1} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \\ a_{j_1,1} & \cdots & a_{j_1,k} \\ \vdots & \ddots & \vdots \\ a_{j_k,1} & \cdots & a_{j_k,k} \end{bmatrix}$$

P jest odpowiednią macierzą permutacji osi współrzędnych przestrzeni $\mathbb{F}_q^n$. Jest to tzw. *standardowa postać bazowa* kodu liniowego *C*.
Dla postaci standardowej kodu *C*

$$B = \begin{bmatrix} I_{k,k} \\ A \end{bmatrix}$$

anihilator (macierz kontrolna) H ma następującą postać:

$$H = [-A \quad I_{n-k,n-k}]$$

gdzie $I_{k,k}$, $I_{n-k,n-k}$ są macierzami jednostkowymi odpowiednich wymiarów.

3. METRYKA HAMMINGA

Definicja. Rozszerzając metrykę dyskretną (jedyną) w ciele $\mathbb{F}_q^n$

$$\| x \| = \begin{cases} 0 & dla\ x = 0 \\ 1 & dla\ x \neq 0 \end{cases}$$

do normy L^1 w przestrzeni $\mathbb{F}_q^n$ wzorem

$$\| v \| = \sum_{j=1}^{n} \| v_j \|, v = (v_1, \ldots, v_n).$$

otrzymujemy tzw. *metrykę Hamminga.*

$$\varrho: \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{Z}_+, \varrho(u, v) = \| u - v \|.$$

Definicja. Dla kodu liniowego $C \subset \mathbb{F}_q^n$ liczbę

$$d = \min_{u,v \in C, u \neq v} \varrho(u, v) = \min_{u,v \in C, u \neq v} \| u - v \| = \min_{v \in C, v \neq 0} \| v \|$$

nazywamy *minimalną odległością Hamminga* lub krócej *odległością Hamminga*[4].

Odległość Hamminga kodu

$$C \subset \mathbb{F}_q^n$$

jest równa minimalnej liczbie liniowo zależnych kolumn anihilatora kodu C.
Dla takiego kodowania możliwe jest wykrycie d błędów kodu C oraz korekcja $\left\lfloor \frac{d-1}{2} \right\rfloor$ tych błędów[4].

4. GENERATOR KODÓW

Generator kodów został zaimplementowany w języku C, przy wykorzystaniu biblioteki algebraicznej Computer Algebra System z Uniwersytetu w Bordeaux.
Wybrano ciało skończone $\mathbb{F}_{7^6}$ rzędu $7^6 = 117\,649$ przyjmując następujące wartości parametrów:

$$p = 7, k = 3, n = 2k = 6,$$

wielomian nieprzywiedlny $f \in \mathbb{F}_7[x]$, stopnia $n=6$:

$$f(X) = X^6 + X^5 + 2X^4 + X^3 + 5X^2 + 3X + 2$$

generator (pierwiastek pierwotny) g ciała $\mathbb{F}_{7^6}$ rzędu 7^6 -1= 117 648:

$$g(X) = 3X^5 + 4X^4 + 5X^2 + 2X + 2$$

Do obliczeń wykorzystano funkcje biblioteki Computer Algebra System (w szczególności wykorzystywane były funkcje *ffinit(),ffgen(), ffprimroot(), fforder()*) :

```
void init_kody(long prec)
{
 GEN p1;
 p = pol_x(fetch_user_var("p"));
 k = pol_x(fetch_user_var("k"));
 n = pol_x(fetch_user_var("n"));
 f = pol_x(fetch_user_var("f"));
 t = pol_x(fetch_user_var("t"));
 g = pol_x(fetch_user_var("g"));
 p = stoi(7);
 k = stoi(5);
 n = gmulsg(2, k);
 f = ffinit(p, gtos(n), -1);
 t = ffgen(f, -1);
 g = ffprimroot(t, NULL);
 p1 = fforder(g, NULL);
 {
  GEN j;
  for (j = gen_0; gcmp(j, p1) <= 0; j =
gaddgs(j, 1))
    pari_printf("%Ps; %Ps\n", j, gpow(g,
j, prec));
 }
 return;
}
```

Realizacją ciała $\mathbb{F}_{7^6}$ jest ucięta algebra wielomianów:

$$\mathbb{F}_{7^6} \simeq \mathbb{F}_7[X]/(f)$$

Obliczenia prowadzone były w uporządkowanej bazie sześciowymiarowej przestrzeni wektorowej $\mathbb{F}_{7^6} \simeq \mathbb{F}_7^6$ nad ciałem $\mathbb{F}_7$:

$$(X^5, X^4, X^3, X^2, X, 1)$$

Macierze automorfizmu Frobeniusa $\sigma: \mathbb{F}_7^6 \to \mathbb{F}_7^6$ oraz inwolucja $\tau = \sigma^3$:

$$\sigma = \begin{bmatrix} 6 & 0 & 2 & 5 & 6 & 0 \\ 4 & 5 & 6 & 4 & 1 & 0 \\ 2 & 3 & 5 & 1 & 3 & 0 \\ 4 & 2 & 1 & 3 & 2 & 0 \\ 2 & 4 & 3 & 6 & 1 & 0 \\ 6 & 6 & 4 & 6 & 2 & 1 \end{bmatrix}, \quad \tau = \begin{bmatrix} 3 & 5 & 5 & 0 & 5 & 0 \\ 6 & 5 & 0 & 3 & 5 & 0 \\ 4 & 4 & 6 & 2 & 1 & 0 \\ 1 & 2 & 5 & 1 & 6 & 0 \\ 1 & 2 & 5 & 2 & 5 & 0 \\ 6 & 2 & 3 & 6 & 2 & 1 \end{bmatrix}$$

operatory rzutu (idempotenty) π^+ i π^-:

$$\pi^+ = \begin{bmatrix} 6 & 1 & 1 & 0 & 1 & 0 \\ 4 & 5 & 0 & 2 & 1 & 0 \\ 5 & 5 & 1 & 6 & 3 & 0 \\ 3 & 3 & 1 & 0 & 4 & 0 \\ 3 & 6 & 1 & 6 & 5 & 0 \\ 4 & 6 & 2 & 4 & 6 & 0 \end{bmatrix}, \quad \pi^- = \begin{bmatrix} 2 & 6 & 6 & 0 & 6 & 0 \\ 3 & 3 & 0 & 5 & 6 & 0 \\ 2 & 2 & 0 & 1 & 4 & 0 \\ 4 & 1 & 6 & 1 & 3 & 0 \\ 4 & 1 & 6 & 1 & 3 & 0 \\ 3 & 1 & 5 & 3 & 1 & 1 \end{bmatrix}$$

bazy podprzestrzeni V^+ i V^- (jako odpowiednie kolumny macierzy):

$$V^+ = \begin{bmatrix} 6 & 1 & 0 \\ 5 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad V^- = \begin{bmatrix} 3 & 1 & 2 \\ 0 & 4 & 5 \\ 6 & 4 & 6 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

oraz elementy $\xi, \xi^{-1} \in V^-$:

$$\xi = 2X^5 + 6X^4 + 5X^3 + 5X^2 + 4,$$

$$\xi^{-1} = 3X^5 + 6X^3 + X^2.$$

Dla ujednolicenia oznaczeń bazę przestrzeni $\mathbb{F}_7^6$ oznaczmy:

$$(X^5, X^4, X^3, X^2, X, 1) = (e_1, e_2, e_3, e_4, e_5, e_6)$$

oraz bazę podprzestrzeni V^+:

$$f_1 = 6e_1 + 5e_2 + e_3$$

$$f_2 = e_1 + e_4 + e_5$$

$$f_3 = e_6$$

Mnożenie w ciele $\mathbb{F}_7^6$ można przedstawić jako tensor

$$\mathbb{F}_7^6 \otimes \mathbb{F}_7^6 \to \mathbb{F}_7^6$$

$$e_j \cdot e_k = \sum_{l=1}^{6} c_{j,k}^{l} e_l, \qquad j,k = 1,\ldots,6$$

gdzie współczynniki $c_{j,k}^{l} \in \mathbb{F}_7$ są stałymi struktury (mnożenia).
Gdy mnożenie przez lewy czynnik ograniczymy do podprzestrzeni $V^+ \in \mathbb{F}_{7^3}$ to otrzymamy częściowy tensor w postaci trzech macierzy:

$$f_1 = \begin{bmatrix} 0 & 6 & 6 & 4 & 6 & 6 \\ 2 & 6 & 5 & 3 & 3 & 5 \\ 3 & 0 & 4 & 6 & 1 & 1 \\ 0 & 2 & 6 & 1 & 5 & 0 \\ 5 & 2 & 4 & 5 & 3 & 0 \\ 2 & 2 & 6 & 2 & 2 & 0 \end{bmatrix}, f_2 = \begin{bmatrix} 1 & 3 & 3 & 6 & 6 & 1 \\ 4 & 4 & 6 & 2 & 5 & 0 \\ 1 & 3 & 3 & 4 & 0 & 0 \\ 6 & 4 & 6 & 2 & 3 & 1 \\ 6 & 0 & 5 & 1 & 4 & 1 \\ 1 & 1 & 2 & 2 & 5 & 0 \end{bmatrix}, f_3 = Id.$$

Włożenie generujące $7^3+1=344$ kodów liniowych:

$$\Theta: \mathbb{P}^1(\mathbb{F}_{7^3}) \to Grass(3,6,\mathbb{F}_7)$$

Realizowane jest następująco:

$$\mathbb{F}_{7^3} \simeq V^+ \ni x \mapsto span_{\mathbb{F}_{7^3}}\{x + \xi\} \subset \mathbb{F}_7^6$$
$$\infty \mapsto V^+ \subset \mathbb{F}_7^6$$

Ogólnie, każda podprzestrzeń jednowymiarowa nad $\mathbb{F}_{7^3}$ jest odwzorowywana na podprzestrzeń wymiaru trzy nad $\mathbb{F}_7$ za pomocą operacji:

$$\mathbb{F}_7^6 \ni u,\ span_{\mathbb{F}_{7^3}}\{u\} \mapsto span_{\mathbb{F}_7}\{f_1 \cdot u, f_2 \cdot u, f_3 \cdot u\}.$$

Poniżej przedstawiono niektóre z funkcji realizujące obliczanie kodów. Funkcja *tuple()* generuje potrzebne krotki, natomiast wywoływana przez nią funkcja *act()* wykonuje odpowiednie operacja macierzowe:

```
void tuple( int n, int k, int d,
 int (*f)( int*, int, int[_n][_k] ), int
*a, int b[_n][_k])
{
 if( d > 0 ) {
        int j;
        for( j=0; j<n; ++j ) {
           a[d-1]= j;
           tuple( n, k, d-1, f, a, b );
        }
 }
 else f( a, k, b );
}

int act( int u[_k])
{
 int v[_n], w[_n], r[_n][_k];
 static unsigned long l= 1UL;
 mulV3( v, Vplus, u );
 addV( w, v, xi0 );
 mul3U( r, w );
 print3U( r );
}
```

W wyniku tych działań otrzymujemy poszukiwane kody:

```
1. [6 5 1 0 0 0
    1 0 0 1 1 0
    0 0 0 0 0 1]

2. [5 2 4 5 4 0
    6 2 6 3 4 0
    2 6 5 5 0 4]

3. [6 0 5 0 6 0
    2 3 2 2 3 6
    1 4 6 5 0 4]

...

344.[2 5 6 4 3 1
     3 4 5 6 0 0
     2 1 4 4 6 3].
```

5. ANALIZA JAKOŚCIOWA KODÓW

Dla rozważanego przykładu możliwe odległości Hamminga wynoszą *1,2,3* oraz *4*:

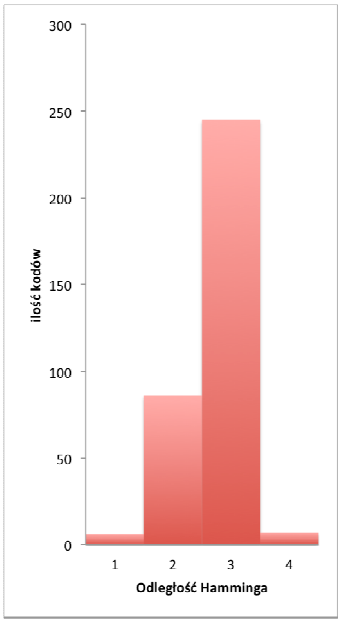
$$1 \le d \le n - k + 1 = 4$$

Ponieważ wyznaczanie odległości Hamminga dla kodów liniowych jest problemem NP-trudnym [5], do obliczenia wag Hamminga wykorzystany został podstawowy algorytm (brute force):

```
int check( int *a, int k, int b[_n][_k]
) {
 int u[_n], j, non0= 0;
 mulV3( u, b, a );
 for( j=0; j < _n; ++j )
    if( u[j] ) ++non0;
```

```
 if( non0> 0 && distHamming > non0 )
   distHamming= non0;
 return non0;
}
```

Rysunek 1 przedstawia odległości Hamminga dla wcześniej wygenerowanych kodów.
Nieznaczna ilość otrzymanych kodów (sześć kodów) jest całkowicie bezużyteczna. Odległość Hamminga wynosi dla nich 1, co nie pozwala nawet na wykrycie błędów. Kolejne osiemdziesiąt sześć kodów również ma niewielkie zastosowanie - odległość Hamminga wynosi dla nich 2, co umożliwia wykrywanie błędów, ale nie pozwala na ich korekcję. Dla większości otrzymanych kodów, dokładnie dla 252, możliwe jest zarówno wykrywanie, jak i korekcja błędów, tj. odległość Hamminga jest większa od 2. Wśród nich uzyskano siedem optymalnych kodów o maksymalnej dla tego przykładu odległości Hamminga wynoszącej 4.



Rysunek. 1 Dystrybucja odległości Hamminga kodów.

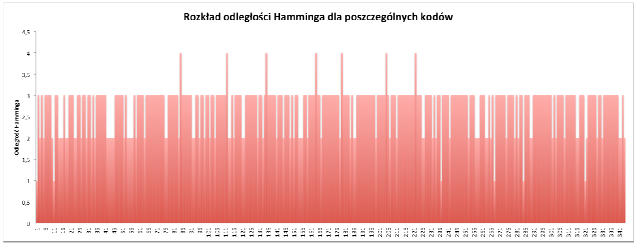
Optymalne kody:

```
   5.[1 6 2 6 5 3
      0 3 2 0 1 5
      0 1 4 2 4 5]


112.[4 3 0 0 6 6
     1 4 5 4 5 3
     5 3 3 6 1 6]


135.[4 3 0 0 6 2
     3 1 3 4 5 5
     0 6 5 3 5 6]
164.[2 3 0 5 4 5
     3 4 5 6 0 1
     3 4 6 0 2 0]
179.[2 3 0 5 4 3
     4 6 4 6 0 2
     4 2 0 2 4 0]
205.[4 2 4 4 3 6
     2 6 4 4 5 1
     3 6 5 6 1 1]
222.[6 5 6 1 0 4
     2 3 2 2 3 0
     2 0 1 1 3 1]
```

Szczegółowy rozkład wyników został zaprezentowany na rysunku 2:



Rysunek. 2 Rozkład minimalnych odległości Hamminga.

Jak widać na omówionym przykładzie opisywana metoda pozwala na generowanie kodów liniowych umożliwiających efektywne korygowanie błędów transmisji. W przyszłości autorzy zamierzają przeprowadzić kompleksowe badania kodów generowanych przy pomocy takiej metody.

Literatura

1. Biswas S., Introduction to Codding Theory: Basic codes and Shannon's theorem. http://www.math.uchicago.edu/~may/VIGRE/VIGRE2008/REUPapers/Biswas.pdf., 2011
2. Frydrych M., Horzelski W., Generator kodów liniowych o skończonych charakterystykach, arXiv 11/2014, CoRR abs/1411., 2014
3. MacWilliams F.J., Sloane N.J.A. , The Theory of Error-Correcting Codes. North-Holland Publishing Company, 1978
4. Pless V., Introduction to the Theory of Error-Correcting Codes. John Wiley and Sons, Inc., 1998
5. Xiao-Yu Hu ,Fossorier, M.P.C., Eleftheriou, E., On the computation of the minimum distance of low-density parity-check codess. Communications, IEEE International Conference, 2004