Un algorithme pour trouver la permutation entre deux codes binaires équivalents

Nicolas Sendrier

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Un algorithme pour trouver la permutation entre deux codes binaires équivalents

Nicolas Sendrier

N° 2853

Avril 1996

THÈME 2

ISSN 0249-6399

INRIA
ROCQUENCOURT

Un algorithme pour trouver la permutation entre deux codes binaires équivalents

Nicolas Sendrier

Thème 2 — Génie logiciel
et calcul symbolique
Projet CODES

Rapport de recherche n° 2853 — Avril 1996 — 10 pages

Résumé : Nous présentons ici un algorithme permettant de retrouver la permutation entre deux codes linéaires binaires équivalents. L'algorithme ne fonctionne que lorsque le groupe d'automorphisme des codes considérés est trivial, c'est-à-dire réduit à la seule identité. Pour deux codes binaires équivalents aléatoires de longueur 1000 et de dimension 500, le temps de calcul varie entre 7 et 80 secondes sur une station de travail DEC 3000/900.

(Abstract: pto)

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : (33 1) 39 63 55 11 – Télécopie : (33 1) 39 63 53 30

An algorithm for finding the permutation between two equivalent binary codes

Abstract: We present here an algorithm able to compute the permutation between two given equivalent binary linear codes. The algorithm only works if the automorphism group of the considered codes is trivial, that is reduced to the identity permutation. For random codes of length 1000 and dimension 500, the computation time varies between 7 and 80 seconds on a workstation DEC 3000/900.

Notations

Soit $\mathbf{F}_q$ le corps fini à q éléments, n un entier strictement positif. Soient $x = (x_1, \ldots, x_n)$ et $y = (y_1, \ldots, y_n)$ deux éléments de $\mathbf{F}_q^n$.

- Pour tout ensemble fini E, nous noterons $|E|$ son cardinal.
- Pour tout entier n nous noterons I_n l'ensemble $\{1, \ldots, n\}$.
- La distance de Hamming entre deux éléments x et y de $\mathbf{F}_q^n$ sera $d_H(x, y) = |\{i \in I_n \mid x_i \neq y_i\}$, le poids de Hamming de x sera $w_H(x) = |\{i \in I_n \mid x_i \neq 0\}$
- Pour tout sous-espace vectoriel C de $\mathbf{F}_q^n$, nous noterons $C^\perp$ son orthogonal, aussi appelé code dual de C, pour le produit scalaire usuel.

1 Introduction

Nous proposons ici un algorithme permettant de déterminer si deux codes linéaires sont équivalents, c'est-à-dire égaux à une permutation près du support, et dans ce cas de donner la permutation.

2 Définitions

2.1 Équivalence

Définition 1 *Soient C et C' deux codes de longueur n. Les codes C et C' seront dits* équivalents *s'il existe une permutation σ de I_n telle que*

$$(x_i)_{1 \leq i \leq n} \in C \Longleftrightarrow (x_{\sigma(i)})_{1 \leq i \leq n} \in C'.$$

Nous noterons $C' \sim C$ ou $C' = \sigma(C)$.

2.2 Codes raccourcis et codes poinçonnés

Définition 2 *Soit C un code linéaire de longueur n, et soit J un sous-ensemble de I_n.*

- *Le code C poinçonné en J est défini par*

$$C_{\widehat{J}} = \{(x_i)_{i \in I_n \setminus J} \mid (x_i)_{1 \leq i \leq n} \in C\},$$

c'est-à-dire que les positions indexées par J sont supprimées.

– *Le code C raccourci en J est défini par*

$$C_{\overline{J}} = \{(x_i)_{i \in I_n \setminus J} \mid (x_i)_{1 \leq i \leq n} \in C \text{ et } \forall j \in J, x_j = 0\},$$

c'est-à-dire que ne seront conservés que les mots du code C dont les coordonnées indexées par J sont nulles.

Dans les deux cas, le nouveau code est de longueur $n - |J|$.

Notons que les deux opérations commutent. Pour tous sous-ensembles disjoints J et J' de I_n nous noterons alors $C_{\widehat{J},\overline{J'}} = C_{\overline{J'},\widehat{J}}$ le code C poinçonné en J et raccourci en J'.

Proposition 1 *Soit C un code linéaire de longueur n, soient J et J' deux sous-ensembles disjoints de I_n et soit σ une permutation de I_n. On a*

$$(C_{\widehat{J},\overline{J'}})^{\perp} = (C^{\perp})_{\overline{J},\widehat{J'}},$$

$$\sigma(C_{\widehat{J},\overline{J'}}) = \sigma(C)_{\widehat{\sigma(J)},\overline{\sigma(J')}}.$$

2.3 Invariants

Soit $\mathcal{L}_n$ l'ensemble des codes linéaires de longueur n sur $\mathbf{F}_q$, soit $\mathcal{L} = \bigcup_{n>0} \mathcal{L}_n$ l'ensemble des codes linéaires sur $\mathbf{F}_q$.

Définition 3 *Un* invariant *sur E est une application $\mathcal{L} \to E$ telle que deux codes équivalents prennent la même valeur.*

Par exemple la longueur, la dimension sont des invariants sur les entiers. L'énumérateur des poids est un invariant sur les polynômes à coefficients entiers.

Corollaire 1 *Soit $\mathcal{V}$ un invariant, soit C un code linéaire de longueur n, soient J et J' deux sous-ensembles disjoints de I_n et soit σ une permutation de I_n. On a*

$$\mathcal{V}(C_{\widehat{J},\overline{J'}}) = \mathcal{V}(\sigma(C)_{\widehat{\sigma(J)},\overline{\sigma(J')}})$$

2.4 Signatures

Définition 4 – *Une* signature *S sur un ensemble F est une application qui à tout code C de longueur n et à tout élément i de I_n associe un élément $S(C,i)$ de F et telle que pour toute permutation σ de I_n et pour tout i dans I_n, $S(\sigma(C),\sigma(i)) = S(C,i)$.*

– *Une signature S est dite* discriminante *pour C s'il existe i et j dans I_n tels que $S(C,i) \neq S(C,j)$.*

– *Une signature S est dite* totalement discriminante *pour C si pour tout i et tout j distincts dans I_n, $S(C,i) \neq S(C,j)$.*

Proposition 2 *Soit C un code de longueur n et soit S une signature totalement discriminante pour C, alors*

1. *le groupe d'automorphisme de C est trivial,*
2. *la signature S est totalement discriminante pour tout code C' équivalent à C.*

En particulier, il n'existe aucune signature totalement discriminante pour un code possédant un groupe d'automorphisme non trivial.

Produit de signature Soient S et S' deux signatures sur E et E' respectivement, la signature produit de S et S', notée $T = S \times S'$, définie par $T(C,i) = (S(C,i), S'(C,i))$ est elle-même une signature sur $E \times E'$.

Signature associée à un invariant Soit $\mathcal{V}$ un invariant sur E, les applications $\widehat{V}$ et $\overline{V}$ sur E définie pour tout code C de longueur n et tout i dans I_n par

$$\widehat{V}(C,i) = \mathcal{V}(C_{\widehat{\{i\}}}) \text{ et } \overline{V}(C,i) = \mathcal{V}(C_{\overline{\{i\}}}),$$

sont des signatures sur E. La signature sur E^2 associée à $\mathcal{V}$ sera le produit de ces deux signatures, noté $V = \widehat{V} \times \overline{V}$.

Partition associée à une signature et un code Soit S une signature sur E, soit C un code linéaire de longueur n. Nous munissons E d'un ordre total noté $<$.

Nous définissons une relation d'équivalence sur I_n par

$$i \ \mathcal{R} \ j \Longleftrightarrow S(C,i) = S(C,j).$$

Soit $\{e_1, \ldots, e_s\} = S(C, I_n)$ avec $e_1 < \ldots < e_s$, l'ensemble des valeurs prises par $S(C, i)$ lorsque i parcourt I_n. Pour tout t, $1 \leq t \leq s$, nous noterons J_t la classe d'équivalence dont les éléments ont pour image e_t. La partition $J_1, \ldots, J_s$, dans cet ordre, de I_n induite par la relation d'équivalence $\mathcal{R}$ sera appelée $(S, C, <)$-partition de I_n. Notons que cette partition est ordonnée, dans le sens où l'ordre des parties $J_1, \ldots, J_s$ n'est pas indifférent.

Proposition 3 *Soit $C' = \sigma(C)$ un code équivalent à C, la $(S, C', <)$-partition de I_n est $\sigma(J_1), \ldots, \sigma(J_s)$.*

Finesse d'une signature Soient S et S' deux signatures sur E et E', la signature S sera dite plus fine que la signature S' pour C si la $(S, C, <_E)$-partition de I_n est plus fine que la $(S', C, <_{E'})$-partition de I_n (les ordres respectifs $<_E$ et $<_{E'}$ sur E et E' ne jouent ici aucun rôle).

2.5 Hull d'un code linéaire

Définition 5 *Le* hull *d'un code linéaire C est défini comme son intersection avec son dual. Nous noterons $\mathcal{H}(C) = C \cap C^{\perp}$.*

Proposition 4 *Soit $\mathcal{V}$ un invariant et C un code linéaire. L'application $\mathcal{V}_{\mathcal{H}} : C \mapsto \mathcal{V}(\mathcal{H}(C))$ est un invariant.*

3 Un algorithme pour trouver la permutation entre des codes équivalents

Soient C et C' deux codes de même longueur. Nous voulons savoir s'il sont équivalents et, dans l'affirmative, calculer une permutation σ telle que $C' = \sigma(C)$.

Dans le cas où C et C' sont équivalents, et que de plus nous connaissons une signature totalement discriminante pour C, l'utilisation de l'Algorithme 1 permet d'obtenir la permutation σ.

Algorithme 1 Soient C et C' deux codes équivalents de longueur n et soit S une signature totalement discriminante pour C.

procédure permutation
entrées: code C,C'; signature S
pour i **dans** I_n

pour j **dans** I_n
si $S(C,j) = S(C',i)$
$\sigma[i] \leftarrow j$
retourner(σ)

Si nous voulons le même résultat sans à priori sur C, C' et S, il faut modifier l'algorithme. Nous noterons $S(C, I_n)$ l'ensemble $\{S(C,i), i \in I_n\}$.

Algorithme 2 Soient C et C' deux codes de longueur n et soit S une signature.

procédure équivalent
entrées: code C,C'; signature S
si $S(C, I_n) = S(C', I_n)$
si $|S(C, I_n)| = n$
$\sigma \leftarrow$ permutation(C,C',S)
si $C' = \sigma(C)$
retourner(σ)
sinon
retourner(NON ÉQUIVALENTS)
sinon
retourner(ÉCHEC)
sinon
retourner(NON ÉQUIVALENTS)

L'Algorithme 2 conclura toujours, sauf dans le cas où $S(C, I_n) = S(C', I_n)$ et $|S(C, I_n)| < n$. Cela signifiera généralement que C et C' sont équivalents, mais que la signature n'est pas assez discrimante pour le calcul de la permutation.

3.1 Raffinement d'une signature discriminante

Soit C un code linéaire, soit une signature S sur E discriminante pour C, et soit $\mathcal{V}$ un invariant sur F. Nous munissons E d'un ordre total noté $<_E$.

Soit $J_1, \ldots, J_s$ la $(S, C, <_E)$-partition de I_n. Puisque S est discriminante, nous avons $s > 1$, d'autre part si $s = n$, alors S et totalement discriminante et il n'est pas nécessaire de poursuivre le calcul. Pour toute partie L de $\{1, .., s\}$, nous noterons $K_L = \bigcup_{l \in L} J_l$, et pour tout $i \in I_n$,

$$S^L_{\mathcal{V}}(C,i) = \begin{cases} \left(\mathcal{V}(C_{\widehat{K_L \cup \{i\}}}), \mathcal{V}(C_{\widehat{K_L}, \overline{\{i\}}}), \mathcal{V}(C_{\overline{K_L}, \widehat{\{i\}}}), \mathcal{V}(C_{\overline{K_L \cup \{i\}}})\right) & \text{si } i \in I_n \setminus K_L \\ \infty & \text{sinon} \end{cases}$$

avec la convention $J_l = \emptyset$ si $l \notin \{1, \ldots, s\}$, l'application $S_{\mathcal{V}}^L$ définit une signature sur $F^4 \cup \{\infty\}$ pour toute partie finie L de $\mathbf{N}^*$. L'élément ∞ sera par convention supérieur à tout élément de F^4 pour tout ordre.

Pour tout s positif, nous munissons l'ensemble des parties de $\{1, .., s\}$ de l'ordre suivant:

$$L \prec_s L' \Longleftrightarrow \begin{cases} |L| < |L'| \\ \text{ou} \\ |L| = |L'| \text{ et } \sum_{l \in L} l < \sum_{l \in L'} l \end{cases}$$

Pour un code C donné dont la $(S, C, <_E)$-partition comporte s parties, nous noterons L le plus petit élément pour $\prec_s$ tel que la signature $T = S_{\mathcal{V}}^L \times S$ soit strictement plus fine que S. La signature T ainsi définie permettra de définir une $(T, C, <)$-partition, où $<$ est un ordre total sur $F^4 \times E$, plus fine que la partition $J_1, .., J_s$.

Chaque fois que nous pourrons obtenir en itérant ce processus une signature totalement discriminante pour C, il sera possible à l'aide des algorithmes décrits plus haut de déterminer si un code donné C' est équivalent à C, et si oui, de donner la permutation entre ces deux codes.

3.2 Une bonne signature

Nous avons vu qu'une signature peut être construite à partir de tout invariant. Cet invariant doit permettre d'obtenir une signature discriminante, ce qui exclus les invariants tels que la longueur ou la dimension. D'autre invariants, tels que la distance minimale, ou mieux encore, la distribution des poids peuvent permettre la construction de signatures discriminantes, toutefois, lorsque la taille du code augmente, ces invariants peuvent s'avérer extrêmement difficiles à calculer.

L'idée est alors de s'appuyer sur la Proposition 4 et d'utiliser un invariant du hull, plus précisément la distribution des poids. Le résultat suivant peut aider à se convaincre qu'un tel invariant pourra être discriminant tout en restant relativement aisé à calculer.

Proposition 5 *[3] La dimension moyenne du hull d'un code linéaire de longueur n sur $\mathbf{F}_q$ tend vers une constante lorsque n tends vers l'infini, cette constante est égale à*

$$\sum_{i>0} \frac{1}{1+q^i}$$

De plus la proportion de codes linéaires de longueur n sur $\mathbf{F}_q$ ayant un hull de dimension donnée l est également une constante lorsque n tends vers l'infini égale à

$$R_l = \frac{R_{l-1}}{q^l - 1} \text{ et } R_0 = \prod_{i \geq 1} \frac{1}{1 + q^{-i}}.$$

Dans le cas binaire ($q = 2$), la dimension moyenne du hull est environ égale à 0.764, et $R_0 = R_1 \approx 0.419$, $R_2 \approx 0.140$, $R_3 \approx 0.020$ ....

L'algorithme décrit dans la section 3 ne peut aboutir pour tout code. En effet, par construction, il ne pourra être appliqué à un code dont le groupe d'automorphisme n'est pas réduit à la seule permutation identité, puisqu'aucune signature totalement discriminante ne peut alors exister (cf. Proposition 2).

De plus, l'algorithme sera impraticable chaque fois que le hull des codes concernés aura une dimension telle que sa distribution des poids ne peut être obtenue par le calcul (dimension supérieure à 40 dans le cas binaire).

Cependant, si nous ne considérons que des codes aléatoires, aucun de ces deux inconvénients n'intervient en pratique. D'abord, dés que la dimension d'un code aléatoire dépasse quelque dizaines d'unités, son groupe d'automorphisme est trivial avec une très grande probabilité. D'autre part, la Proposition 5 nous assure qu'un code choisi aléatoirement n'a presque aucune chance de posseder un hull de grande taille.

3.3 Temps de calcul

L'algorithme décrit plus haut a été programmé en langage C et exécuté sur une station de travail DEC 3000/900. Les temps de calculs obtenus sont donnés pour diverses tailles dans la Table 1.

Références

[1] Leon (J.S.). – Computing automorphism groups of error-correcting codes. *IEEE Transactions on Information Theory*, vol. 28, n° 3, mai 1982, pp. 496–511.

[2] Sendrier (N.). – On the structure of a randomly permuted concatenated code. *In : Livre des résumé – EUROCODE 94*, éd. par Charpin (P.). INRIA, pp. 169–173. – Abbaye de la Bussière sur Ouche, France, octobre 1994.

n	k	nombre d'essais	temps (en secondes)		
			moyen	minimum	maximum
1000	500	120	13.2	7.1	79.9
1000	200	100	27.2	7.0	700.6 [a]
500	250	100	2.3	1.1	10.2
500	100	100	2.7	1.3	31.7
250	125	100	0.59	0.30	4.6
250	50	100	0.49	0.31	2.7
100	50	100	0.16	0.13	0.40

Table 1 - *Temps de calcul pour des codes binaires*

[a] deux des 100 essais on dépassé 10 minutes, le maximum des 98 autres est de 45.7 secondes, et leur moyenne est de 13.2 secondes

[3] Sendrier (N.). – *On the dimension of the hull.* – Rapport de Recherche n° 2862, INRIA, octobre 1995.

[4] Sendrier (N.). – *On the structure of a randomly permuted concatenated code.* – Rapport de Recherche n° 2460, INRIA, janvier 1995.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399