Bases de Gröbner et Codes Cycliques

On the Decoding of Cyclic Codes Using Gröbner Bases

Philippe Loustaunau, Eric V. York

Damien Stehlé

Lundi 11 Mars 2002

- Introduction
- De l'équation de décodage à un système d'équations polynômiales
- Utilisation des bases de Gröbner
- L'algorithme FGLM
- Un exemple de décodage
- Conclusion

Introduction:

- Cooper (1990): Bases de Gröbner et décodage des codes BCH

- Chen, Reed, Helleseth, Troung (1994): Bases de Gröbner et codes cycliques

Malheureusement il y a une erreur dans leur article, et ils utilisent l'algorithme de Buchberger: c'est super-exponentiel !!!

- Loustaunau et York (1997): Clarification et gain de vitesse

1. De l'équation de décodage à un système d'équations polynômiales:

Soit $\mathcal{C}$ un code cyclique [n,k,d] sur $\mathbf{F}_q$, avec $(n, q) = 1$.
α une racine primitive n-ième de l'unité.
$g(X)$ le polynôme générateur de $\mathcal{C}$,
r =deg$(g) = n - k$, et $\alpha^{i_1}, ..., \alpha^{i_r}$ ses racines,
t $(= \lfloor \frac{d}{2} \rfloor)$ le nombre d'erreur que l'on veut corriger.

Décoder $(y_0, y_1, ..., y_{n-1})$, c'est résoudre:

$$\begin{bmatrix} 1 & \alpha^{i_1} & \alpha^{2i_1} & ... & \alpha^{(n-1)i_1} \\ 1 & \alpha^{i_2} & \alpha^{2i_2} & ... & \alpha^{(n-1)i_2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{i_r} & \alpha^{2i_r} & ... & \alpha^{(n-1)i_r} \end{bmatrix} \cdot \begin{bmatrix} e_0 \\ e_1 \\ \vdots \\ \vdots \\ e_{n-1} \end{bmatrix} = \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ \vdots \\ s_{n-1} \end{bmatrix}$$

où $(s_0, s_1, ..., s_{n-1})$ est le syndrome.

Pour cela, définissons les polynômes:

- $f_j = y_1 z_1^{i_j} + y_2 z_2^{i_j} + ... + y_t z_t^{i_j} - x_j$ pour $j = 1..r$
- $h_k = z_k^{n+1} - z_k$ pour $k = 1..t$
- $l_k = y_k^{q-1} - 1$ pour $k = 1..t$

Remarques:

1. $x_j \to s_j$, $z_k \to \alpha^{l_k}$, et $y_k \to e_k$
2. En caractéristique 2, on peut enlever les y_k.
3. Prendre $k = 1..t'$ avec $t' > t$, on en reparlera.
4. Rajouter les $m_{k,l} = z_k z_l \frac{z_k^n - z_l^n}{z_k - z_l}$ n'apporte rien.

On s'intéresse aux zéros de $I =< f_j, h_k, l_k >$: $\mathcal{V}$.
p= $(s_1, .., s_r || 0, .., 0, \alpha^{l_1}, .., \alpha^{l_{t-\tau}} || \times, .., \times, \beta_1, .., \beta_\tau)$
Les l_i sont les positions de l'erreur, et les β_k les coefficients correspondants.
Il y a des zéros non valides: collision possible des l_k.

Si $\mathcal{V}|_i$ est la projection de $\mathcal{V}$ sur les i premières coordonnées,

Théorème:

Il y a τ erreurs dans **y** si et seulement si:
pour $k \leq t - \tau$, (**s**,0^k) $\in \mathcal{V}|_{r+k}$ et (**s**,$0^{t-\tau+1}$) $\notin \mathcal{V}|_{r+t-\tau+1}$
De plus, $\{l_i / (\mathbf{s}, 0^{t-\tau}, \alpha^{l_i}) \in \mathcal{V}|_{r+t-\tau+1}\}$ est la localisation de l'erreur.

2. Utilisation des bases de Gröbner:

Sur $\mathbf{F}[X_1, .., X_n]$, on introduit l'ordre lexicographique:
1. $X_1 < X_2 < .. < X_n$,
2. $\prod_{i=1}^{n} X_i^{\alpha_i} < \prod_{i=1}^{n} X_i^{\beta_i}$ si $\alpha_1 > \beta_1$ ou $\alpha_1 = \beta_1$ et $\alpha_2 > \beta_2$,...
3. $P[X_1, .., X_n] > Q[X_1, .., X_n]$ si $lt(P) > lt(Q)$

Une base de Gröbner $G = \{g_1, ..g_k\}$ d'un idéal I de $\mathbf{F}[X_1, .., X_n]$ doit:
générer I et vérifier $< lt(g_1), .., lt(g_k) >=< lt(I) >$.

Remarques:
1. Existence des bases de Gröbner (Buchberger)
2. Complexité super-exponentielle

Théorème:

Soit G une base de Gröbner de I et $m < n$.
Alors $G_m = G \cap \mathbf{F}[X_1, .., X_m]$ est une base de Gröbner de $I \cap \mathbf{F}[X_1, .., X_m]$

Autres faits importants:

1. Si $\mathcal{V}(I)$ est l'ensemble des zéros de I, $\mathcal{V}(I)|_m = \mathcal{V}(I \cap \mathbf{F}[X_1, .., X_m])$.

2. Si $G \subset \mathbf{F}[X_1, .., X_n]$ est une base de Gröbner de I, si $x_1, .., x_m \in \mathbf{F}$, alors $G(x_1, .., x_m, X_{m+1}, .., X_n)$ est une base de Gröbner de $I(x_1, .., x_m, X_{m+1}, .., X_n)$.

3. $G \subset \mathbf{F}[X]$ est une base de Gröbner de $< P_1, .., P_k >$ si et seulement si elle contient le pgcd des P_i.

D'où le théorème de décodage:

Théorème:

Il y a τ erreurs dans **y** si et seulement si:
$\forall k \leq t-\tau$, $\forall g \in G_{r+k}$, $g(\mathbf{s},0^k) = 0$
et $\exists g \in G_{r+t-\tau+1}$ tel que $g(\mathbf{s},0^{t-\tau+1}) \neq 0$.

De plus, si $G_{r+t-\tau+1} = \{g_1, .., g_k\}$,
$< g_1(\mathbf{s},0^{t-\tau}, X), .., g_k(\mathbf{s},0^{t-\tau}, X) > \subset \mathbf{F}_q[X]$ est engendré par le polynôme localisateur de l'erreur, et celui-ci fait partie des $g_i(\mathbf{s},0^{t-\tau}, X)$.

Rappel: Le polynôme localisateur de l'erreur est:
$$\prod_{i/e_i \neq 0}(X - \alpha^i)$$

Il y a 2 phases de décodage: un long précalcul, qui peut être fait une fois pour toutes, et le décodage lui-même, rapide, à partir du syndrome.

Précalcul:

- Trouver une base de Gröbner des f_j, h_k, l_k
- Construire les G_i pour $i = r..(r+t)$ en ne gardant que les polynômes ne contenant pas les plus grandes variables
- Construire les $G'_i = G_i(x_1, .., x_r, 0^{i-r})$ et les $G''_i = G_i(x_1, ..x_r, 0^{i-r-1}, X)$

Décodage:

- Calculer le syndrôme (multiplication matrice-vecteur: $O(n^2)$)
- Calculer les $G'_i(\mathbf{s})$ jusqu'à en trouver un non nul: $i = i_0$
- Trouver le localisateur de l'erreur dans $G''_{i_0}(\mathbf{s})$
- Factoriser le localisateur de l'erreur
- Corriger l'erreur

Fait: Plus il y a d'erreurs, plus c'est rapide!

3. L'algorithme FGLM:

Auteurs: Faugere, Gianni, Lazard, Mora

Ici, FGLM suffit, et c'est plus rapide que Buchberger...

Input: Une base de Gröbner $\{g_1, ..g_k\}$ de I pour un ordre lexicographique $<_1$
Output: Une base de Gröbner de I pour un ordre lexicographique $<_2 \neq <_1$

Dans notre cas, $\{f_i, g_k, h_k\}$ est une base de Gröbner pour l'ordre lexicographique: $y < z < x$, et on voudrait l'ordre lexicographique $x < z < y$.

Soit $\mathcal{R}$ la fonction "reste modulo I pour $<_1$".
C'est un morphisme de $\mathbf{F}[X_1, .., X_n]$ dans $\mathbf{F}[X_1, .., X_n]/I$.
Soient $M_0 <_2 M_1 <_2 .. <_2 M_i <_2 ...$ les monomes.

Fait: Un polynôme $\sum_{j=0}^{i} \sigma_j M_j$ de terme dominant M_i (pour $<_2$) est dans I si et seulement si:
$$\mathcal{R}(\sum_{j=0}^{i} \sigma_j M_j) = \sum_{j=0}^{i} \sigma_j \mathcal{R}(M_j) = 0.$$

$\rightarrow$ Algèbre linéaire!
Comme les $\mathcal{R}(X_j)$ sont en nombre fini tout va bien.

Input : G base de Gröbner pour $<_1$
Output : G' base de Gröbner pour $<_2$

G':=[]; Nexts:=[]; M:=1; Basis:=[];
/* Basis est une base de $\mathbf{F}[X_1, .., X_n]/I$ */

Tant que $M \neq 0$ faire
Si M n'est pas dans $< lt(G') >$,
Alors calculer $\mathcal{R}(M)$;
S'il existe $\{\sigma_j\}$ / $\mathcal{R}(M) = \sum_{M_j \in Basis} \sigma_j \mathcal{R}(M_j)$
Alors, mettre $M - \sum_{M_j \in Basis} \sigma_j M_j$ dans G'
/* on a trouvé un nouvel élément! */
Sinon, mettre M dans Basis,
InsertNexts(M)
/* Rajoutter $(x_1 M, .., x_r M, z_1 M, .., z_t M, y_1 M, .., y_t M)$ */

M :=pop(InsertNexts)

1. Comme ce sont les G_k qui nous intéressent, on peut enlever les y_j de InsertNexts.
2. Complexité: $O(|var|.|I|^3) = O((n+1)^{3t+1})$.
3. Prendre $q = 2$ simplifie pas mal de choses.

4. Un exemple de décodage:

Code de Golay [23,12,7], avec $t = 3$ et $q = 2$.
$g(X)$ a pour racine α (racine primitive 23^e de l'unité),
(BCH non primitif de distance construite 3)

D'où: $x_1 + z_1 + z_2 + z_3, z_1^{24} + z_1, z_2^{24} + z_2, z_3^{24} + z_3$
$G_0 = \{x_1^{2048} + x_1\}$
$G_1 = G_0 \cup \{z_1 + z_1^{24}, z_1^3 + ...\}$
$G_2 = G_1 \cup \{z_2 + z_2^{24}, z_2^2 z_1 + ..., z_2^3 x_1^{24} + ...\}$
$G_3 = G_2 \cup \{z_3 + z_2 + z_1 + x_1\}$

$G'_0 = G''_0 = G_0$
$G'_1 = G_0 \cup \{x_1^{1338} + x_1^{1292} + ... + x_1^4\}$
$G''1 = G_1$
$G'_2 = G'_1 \cup \{x_1^{256} + x_1^3, x_1^{1337} + x_1^{1291} + ... + x_1^{26}\}$
$G''_2 = G'1 \cup \{z^{24} + z, zx_1 + ... + x_1^3, zx_1^{24} + ... + x_1^{26}\}$
$G'_3 = G'_2 \cup \{x_1\}$
$G''_3 = G'_2 \cup \{z + x_1\}$

0 erreur : **s**=0
3 erreurs : G''_1 donne le polynôme localisateur
2 erreurs: G''_2 donne deux possibilités, de degré au plus deux chacune. On choisit $z^2 x_1 + z x_1^2 + x_1^2 56 + x_1^3$
1 erreurs: G''_3 donne le polynôme localisateur.

Conclusion

C'est mieux que Buchberger, mais on ne risque pas d'aller très loin pour autant!
Quelle est la complexité du décodage?

Questions:

1. Peut-on choisir d'autres polynômes?
2. Peut-on raisonner modulo $x^\lambda = x$?
3. Peut-on faire plus efficace qu'en prenant FGLM?
4. Existe-t-il une variante de FGLM plus rapide?
5. Peut-on corriger plus de t erreurs?