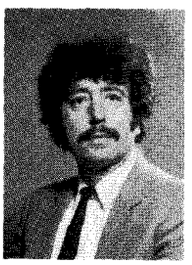
Étude comparative de logiciels de factorisation de polynômes binaires

Experimental comparison between softs polynomial factorization



Alain POLI

Laboratoire AAECC, Université P.-Sabatier, 31000 TOULOUSE

Professeur à l'IUT Informatique de l'Université P.-Sabatier de Toulouse. Diplômes : thèse de 3e cycle et thèse d'état. Directeur du AAECC, laboratoire LSI, domaine : les codes correcteurs polynomiaux et leurs applications.

C. RIGONI

Laboratoire AAECC, Université P.-Sabatier, 31000 TOULOUSE

Étudiant 3e cycle, boursier DGRST, à l'Université P.-Sabatier de Toulouse. Diplôme : DEA Informatique. Membre du AAECC, laboratoire LSI, domaine : les codes correcteurs polynomiaux et leurs applications.

RÉSUMÉ

Nous présentons essentiellement des résultats expérimentaux concernant trois logiciels opérant sur deux ordinateurs : un B6700 de Burroughs et un DPS 8/70 de CII-HB.

MOTS CLÉS

Factorisation de polynômes, logiciel de factorisation.

SUMMARY

We essentially present some experimental results which concern three softwares implemented on two computers: a B6700 and a DPS 8/70.

KEY WORDS

Polynomial factorization, soft for factorization.

TABLE DES MATIÈRES

Introduction

Une des activités du AAECC (Algèbre Appliquée Et Codes Correcteurs) est la production de logiciels performants dans les domaines des codes correcteurs, de la simulation de transmission et de la factorisation de polynômes.

Les logiciels que nous avons mis au point sont opérationnels sur divers ordinateurs et microordinateurs : TRS I, TRS III, Apple III, B6700, DPS 8/70, CDC 750. Les langages de programmation utilisés sont : Basic, Fortran IV, Fortran V, Algol, Pascal.

Nous présentons dans ce qui suit trois de nos logiciels qui permettent de factoriser des polynômes binaires de grands degrés (jusqu'à 2100), en utilisant l'algorithme bien connu de Berlekamp (réf. [1]).

Précisons bien que nous ne donnons pas ici une nouvelle méthode performante de factorisation. Le lecteur intéressé par les complexités théoriques de problèmes de factorisation est renvoyé à [5].

Soit $f(X)$ un polynôme binaire. Nous allons décrire deux algorithmes de factorisation de $f(X)$.

Dans une première partie nous situons le cadre algébrique.

Dans une seconde partie nous décrivons sommairement les algorithmes mis en œuvre.

Dans la dernière partie nous donnons les résultats expérimentaux obtenus.

Partie I. Le cadre algébrique

Nous donnons quelques propriétés algébriques très élémentaires permettant de fixer le cadre algébrique précis dans lequel nous travaillons pour factoriser $f(X)$.

Propriété 0 : Il est possible de toujours se ramener à la factorisation d'un polynôme à racines simples.

Démonstration : En effet soit $g(X)$ le polynôme binaire suivant :

$$g(X) = p_1^{m_1}(X) \times \ldots \times p_N^{m_N}(X),$$

où $p_i(X)$ est irréductible $(1 \leqq i \leqq N)$. Où $m_1, \ldots, m_t$ sont les multiplicités paires.

Alors nous factorisons :

$$\frac{g(X)}{\text{pgcd}(g(X), g'(X))} = f(X) = p_{t+1}(X) \times \ldots \times p_N(X),$$

où $g'(X)$ est la dérivée de $g(X)$.

$f(X)$ est un polynôme à racines simples.

On factorise alors $f(X)$ pour obtenir $p_{t+1}(X), \ldots, p_N(X)$.

Reste alors à factoriser le polynôme $h(X)$:

$$h(X) = p_1^{m_1}(X) \times \ldots \times p_t^{m_t}(X).$$

Pour cela il suffit d'extraire les multiplicités puissances de 2 de $h(X)$, et d'appliquer à nouveau ce que nous avons indiqué ci-dessus.

Soit donc $f(X)$ un polynôme binaire à racines simples :

$$f(X) = p_1(X) \times \ldots \times p_N(X),$$

où $p_i(X)$ est irréductible $(1 \leqq i \leqq N)$.

Énonçons quelques propriétés de l'algèbre A égale à :

$$A = \mathbb{F}_2[X]/(f(X)).$$

Propriété 1 : A est principale. Pour tout $a(X)$ de A on a :

$$(a(X)) = (\text{pgcd}(f(X), a(X))).$$

Nous appelons idempotent de A tout élément $e(X)$ vérifiant :

$$e^2(X) = e(X).$$

Soit h l'endomorphisme de A défini par :

$$h : \quad a(X) \to a^2(X) \qquad (\forall a(X) \in A).$$

Propriété 2 : L'ensemble des idempotents de A est le sous-espace vectoriel propre de h associé à la valeur propre 1.

Ce sous-espace est classiquement appelé espace de Berlekamp.

Propriété 3 : L'élément $\lambda_i(X).(f(X)/p_i(X))$ de A est un idempotent noté $e_i(X)$ pour $\lambda_i(X)$ défini par :

$$\lambda_i(X).\frac{f(X)}{p_i(X)} \equiv 1 \qquad (\text{modulo } p_i(X)).$$

$e_i(X)$ est l'idempotent générateur de l'idéal minimal $(f(X)/p_i(X))$. On dit que e_i est un idempotent primitif de A.

Propriété 4 : On a :

$$A = (e_1) \oplus \ldots \oplus (e_N)$$

et

$$A \simeq B = \mathbb{F}_2[X]/(p_1(X)) \times \ldots \times \mathbb{F}_2[X]/(p_N(X)).$$

L'isomorphisme φ_N de A dans B est défini par :

$$\varphi_N : \quad a_1 e_1 + \ldots + a_N e_N \rightarrow (a_1, \ldots, a_N).$$

Propriété 5 : Soit $e(X)$ un élément de A.
Si : $e(\alpha_i) = 1$ pour toute racine α_i de $p_i(X)$ et si : $e(\alpha_j) = 0$ pour toute racine α_j de $f(X)/p_i(X)$, alors $e(X)$ est égal à l'idempotent $e_i(X)$.
Soit π_k la projection dans B définie par :

$$\pi_k : \quad (a_1, \ldots, a_N) \rightarrow (a_1, \ldots, a_k, 0, \ldots, 0).$$

Désignons par I_k l'isomorphisme défini de $\pi_k(B)$ dans B_k ($B_k = \mathbb{F}_2[X]/(p_1(X) \times \ldots \times \mathbb{F}_2(X]/(p_k(X)))$ par :

$$I_k : \quad (a_1, \ldots, a_k, 0, \ldots, 0) \rightarrow (a_1, \ldots, a_k).$$

Soit enfin φ_k l'isomorphisme de A_k ($A_k = \mathbb{F}_2[X]/(p_1(X) \times \ldots \times p_k(X))$) dans B_k, défini de la même façon que φ_N dans la propriété 4.

Propriété 6 : Le morphisme surjectif $\varphi_k^{-1} \circ I_k \circ \pi_k \circ \varphi_N$ de A dans A_k peut être défini par :

$$a(X) \rightarrow \bar{a}(X),$$

où $\bar{a}(X)$ est le reste de la division de $a(X)$ par $p_1(X) \times \ldots \times p_k(X)$.
Démonstration : Considérons dans A l'idempotent primitif $e_i (1 \leqq i \leqq k)$.
Écrivons l'égalité d'Euclide :

$$e_i(X) = q(X) . p_1(X) \times \ldots \times p_k(X) + r_i(X).$$

On note que l'on a :

$$r_i(\alpha_i) = e_i(\alpha_i) = 1,$$

pour toute racine α_i de $p_i(X)$,

$$r_i(\alpha_j) = e_i(\alpha_j) = 0,$$

pour toute racine α_j de $(p_1(X) \ldots p_k(X))/p_i(X)$.

Donc $r_i(X)$ est l'idempotent primitif de A_k générateur de l'idéal minimal $(p_1(X) \ldots p_k(X)/p_i(X))$.
Si l'on considère un élément $a(X)$ de A, on a :

$$a(X) = a_1(X) . e_1 + \ldots + a_N(X)\, e_N.$$

D'où l'on déduit que :

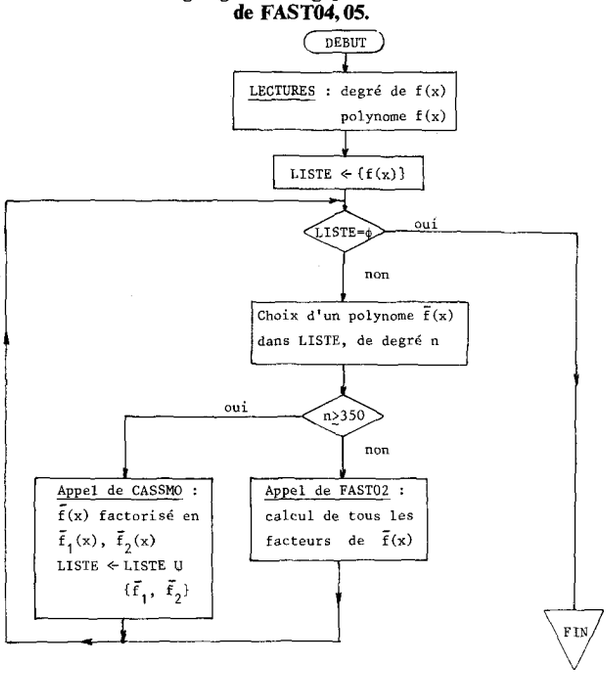
$$\bar{a}(X) = a_1(X)\, r_1 + \ldots + a_k(X)\, r_k.$$

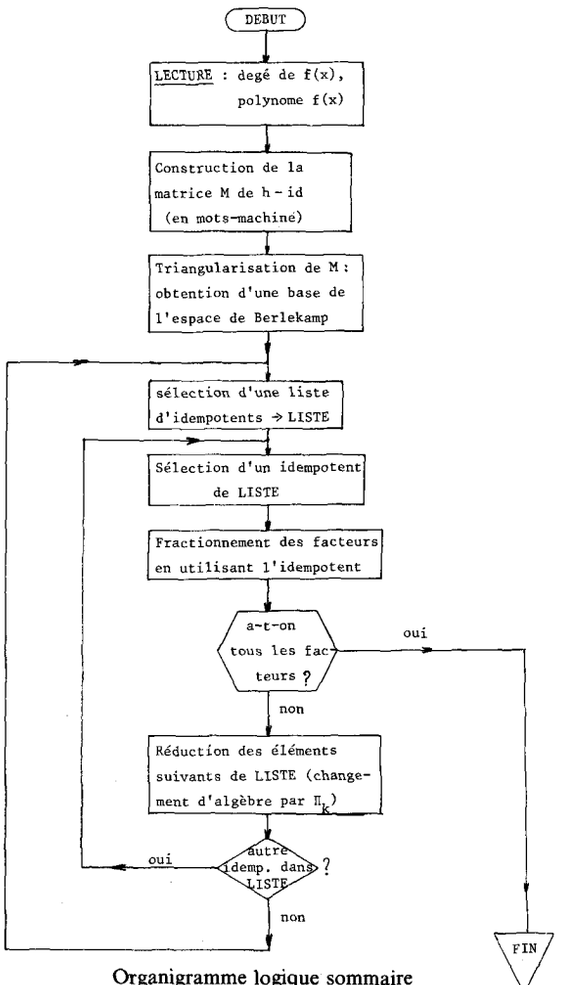
Car le degré de $a_i(X)$ est inférieur au degré de $p_i(X) . (1 \leqq i \leqq k)$.
Ceci achève la preuve de la propriété 6.
Nous décrivons maintenant la factorisation de $f(X)$. A partir d'une base de l'espace de Berlekamp, on possède au moins un idempotent non trivial (ni nul, ni égal à un).
Cet élément $e(X)$ factorise $f(X)$ en deux facteurs $f_1(X)$ et $f_2(X)$ car $\varphi_N(e(X))$ a au moins une composante nulle.
On peut alors supposer que : $f_1(X) = p_1(X) \times \ldots \times p_k(X)$.
D'après la propriété 6 on sait construire un système de générateurs de l'espace de Berlekamp de A_k. On peut ensuite itérer.
Notons que le rôle des idempotents n'est pas essentiel. Ce qui importe est d'arriver à construire un élément $a(X)$ non nul dont la représentation dans B possède au moins une composante nulle. Toutes les techniques visent ce but.

Partie II. Description sommaire des algorithmes proposés

Nous présentons maintenant deux algorithmes de factorisation, l'un implanté sur deux ordinateurs : le B6700 (nom du programme : FAST04) et le DPS 8/70 du CICT de Toulouse (nom du programme : FAST05).
L'autre algorithme est implanté sur un B6700, nom du programme : FAST06.
L'algorithme programmé dans FAST04 et FAST05 est l'algorithme classique de Berlekamp pour des polynômes de degré inférieur à 350. La complexité théorique de cette partie est donc en $O(n^3)$ qui est la complexité de la méthode de Berlekamp pour des polynômes binaires. Nous indiquons que pour de petites valeurs de p la méthode de Berlekamp est la meilleure connue.
Lorsque le degré des polynômes est compris entre 350 et 2 100, nous avons choisi de factoriser $f(X)$ en deux facteurs $f_1(X)$ et $f_2(X)$. La difficulté dans ce cas réside dans le fait que pour construire l'espace de Berlekamp, il est nécessaire de construire une matrice de taille : $n \times n$, si n est le degré du polynôme. Soit par un polynôme de degré 2 100, la taille de la matrice est : $2\,100 \times 2\,100$, ce qui n'est pas concevable sur le B6700, ni sur le DPS 8/70. Aussi utilisons-nous ici le fait que les polynômes sont à coefficients binaires. Ceci permet la simplification suivante : plusieurs coefficients binaires sont stockés dans une variable numérique. Ceci permet alors d'envisager la factorisation de polynômes binaires de grands degrés.
Toutefois les performances de cette technique sont directement liées à la taille du mot machine qui est le registre binaire alloué à chaque variable numérique. Lorsque nous utiliserons cette technique nous dirons que l'on factorise dans les mots machine.
Le sous-programme qui factorise les polynômes de degrés inférieurs à 350 s'appelle FAST02.
Le sous-programme qui décompose en produit de deux facteurs les polynômes de degré supérieur à 350 s'appelle CASSMO.

Organigramme logique sommaire de FAST04, 05.





Organigramme logique sommaire de FAST06.

L'algorithme programmé dans FAST06 est également l'algorithme de Berlekamp, toutefois toutes les fonctions du logiciel sont cette fois programmées dans les mots machine. La technique est la même quel que soit le degré du polynôme.

Nous indiquons que les logiciels que nous avons réalisés n'utilisent pas les supports de mémoire de masses. (Sauf pour les opérations de swapping inhérentes à certains systèmes d'exploitation.)

LMOC, la taille du mot machine est variable d'un ordinateur à l'autre :

– sur le B6700, LMOC est égal à 37 positions binaires;

– sur le DPS 8/70, LMOC est égal à 34 positions binaires.

Un autre intérêt de ces trois logiciels est que LMOC dans chacun des programmes est une variable. Ceci assure la portabilité des logiciels sur d'autres ordinateurs que ceux mentionnés.

Partie III. Résultats expérimentaux

Dans cette partie nous donnons des résultats expérimentaux concernant chacun des trois logiciels : FAST04, FAST05 et FAST06.

Le nombre moyen de polynômes testés par degré est 10.

Les temps sont donnés en minutes, secondes.

degré de f(X)	Temps moyen de calcul pour FAST04	FAST05	FAST06
40	2"9	1"1	0"7
60	4"1	1"3	1"
80	4"7	1"8	1"6
100	7"2	2"6	2"3
150	14"3	5"4	4"9
200	30"8	11"3	8"2
250	51"7	20"2	14"2
300	1'24"	32"	22"4
350	2'00"	44"9	29"7
400	2'27"	1'12"	42"7
450	2'45"	2'33"	1'3"
500	4'03"	3'01"	1'15"
600	4'13"		2'07"
700	7'51"		3'05"
800	9'59"		4'31"
900		DPS 8/70	6'18"
1000			8'21"
1250			15'39"
1500			26'42"
1750	B 6700		40'06"
2000			59'41"

Temps de calcul des polynômes de la forme : $X^n - 1$.

Polynome	Temps de calcul pour FAST04	FAST05	FAST06
$X^{63}-1$	3" 6	1" 3	1" 4
$X^{127}-1$	7" 3	2" 6	3" 2
$X^{255}-1$	34" 4	7" 8	12"
$X^{511}-1$	1'20"	20" 3	30" 9
$X^{2047}-1$	14' 9"	5'48"	7'16"

COMPLEXITÉ EXPÉRIMENTALE

Soit un polynôme $f(X)$ de degré d. Soit t le temps (en secondes) nécessaire à sa factorisation. L'ensemble des couples ($\log d$, $\log t$) forme un nuage de points que l'on peut ajuster par les moindres carrés à une droite d'équation :

$$\log t = a \log d + b \qquad (\log : \text{en base } 10).$$

On en déduit alors que :

$$t = 10^b . d^a.$$

Cette formule donne la complexité expérimentale.

Pour d inférieur ou égal à 300 on a :

- pour FAST04 : $3{,}3 . 10^{-3} . d^{1{,}73}$;
- pour FAST05 : $9{,}5 . 10^{-4} . d^{1{,}78}$;
- pour FAST06 : $8{,}1 . 10^{-4} . d^{1{,}76}$.

Conclusion

L'étude présentée ici est une simple comparaison expérimentale entre plusieurs logiciels appliquant deux versions différentes de l'algorithme de Berlekamp. Les variations par rapport à l'algorithme de Berlekamp initial sont liées à des exploitations diverses de la donnée de la base de Berlekamp.

Toutefois nous indiquons que nos logiciels permettent de factoriser des polynômes binaires de grands degrés (≈ 2000) en utilisant les mots machines. On notera donc l'importance du choix de l'ordinateur. En effet la taille et les temps d'exécution des instructions de manipulation des mots machines varient sensiblement d'un calculateur à l'autre.

BIBLIOGRAPHIE

[1] E. R. BERLEKAMP, *Algebraic coding theory*, MacGram-Hill, 1968.

[2] P. CAMION, Un algorithme de construction des idempotents primitifs d'idéaux d'algèbres sur $\mathbb{F}_q$, *C. R. Acad. Sc.*, Paris, 290, 1980, p. 291.

[3] A. POLI, Un logiciel de factorisation de polynômes binaires : FAST02, *Revue du CETHEDEC*, NS 74, 1983, p. 127-136.

[4] D. COPPERSMITH et S. WINOGRAD, On the asymptotic complexity of matrix multiplication, *SIAM J. Comp.*, 11, n° 3, août 1982, p. 472-492.

[5] D. LUGIEZ, Algorithmes de factorisation de polynômes, *Thèse de* 3e *cycle*, INP, Grenoble, 1984.