Probabilistic Computation of the Smith Normal Form of a Sparse Integer Matrix

Mark Giesbrecht

1996

Le calcul de la forme normale de Smith d'une matrice à coefficients entiers a de nombreuses applications, par exemple dans le domaine de l'algèbre [1]. L'une des méthodes pour le faire est le calcul des pgcd des mineurs de la matrice.

L'article propose un algorithme probabiliste Monte Carlo permettant de calculer ces pgcd pour une matrice creuse. Cet algorithme, se basant sur de nouvelles propriétés étudiées par Giesbrecht dans [2], est plus rapide que le meilleur algorithme déterministe alors connu, et moins coûteux en mémoire. De plus, il utilise le concept de "boîte noire", ce qui permet de tenir compte du caractère creux de la matrice.

Dans une deuxième partie, l'auteur développe une amélioration de l'un des points de cet algorithme, qui permet d'étudier plus facilement sa convergence. Ce deuxième algorithme est également de type Monte Carlo.

1 Boîte noire

L'un des intérêts majeurs des algorithmes proposés est le concept de boîte noire : il n'y a pas besoin de connaître directement la matrice A dont on cherche à calculer une forme normale de Smith.

Dans le premier algorithme, il suffit d'avoir une boîte noire qui, pour tout vecteur v, calcule Av. Ainsi, si cette boîte noire exploite le caractère creux de A (pas d'ajout de coefficients initialement nuls, pas de croissance des coefficients), on exploite également la structure creuse de A dans notre algorithme.

Dans le deuxième algorithme, il suffit d'avoir une boîte noire permettant l'évaluation d'un uplet de polynômes, sans avoir besoin de connaître ces polynômes explicitement. Cela est utile de manière générale car des polynômes sont coûteux en mémoire, et plus particulièrement dans notre exemple, car il est également coûteux en temps de calculer les polynômes dont on a besoin.

Cet article est donc intéressant car il parvient à exploiter le concept de boîte noire en particulier pour le calcul du déterminant, ce qui n'avait encore jamais été fait.

2 Premier algorithme

2.1 Algorithme

Cet algorithme, `DeterminantalDivisors`, calcule le rang r et, pour tout k, le pgcd d_k des mineurs d'ordre k d'une matrice creuse $A \in \mathcal{M}_{m,n}(\mathbb{Z})$ avec probabilité supérieure ou égale à $1-\epsilon$. À partir de ces pgcd, il est immédiat de calculer la forme normale de Smith (voir annexe A). On ne connaît pas A, mais seulement une boîte noire qui, pour tout vecteur v, calcule Av.

Voici l'algorithme `DeterminantalDivisors` :

1. Poser $r = 0; d_1, \ldots, d_m = 0; count = 0$
2. Tant que $count <= \log\left(\frac{1}{\epsilon}\right) + 3\log(m) + \log(\log(A))$ faire
3. Choisir au hasard des matrices U Toeplitz supérieure, L Toeplitz inférieure et D diagonale, à coefficients dans $\mathbb{Z}$
4. Poser $B = UALD$ et définir une boîte noire qui, pour tout vecteur v, calcule Bv
5. Calculer le polynôme caractéristique $g = x^s + \sum_{k=0}^{s-1} b_k x^k$ de B avec forte probabilité
6. Poser $k = \text{Min}\{i \in \mathbb{N}, b_i \neq 0\}$
7. Si $k > 1$ ou ($k = 0$ et $s < m$)
8. Alors retourner à 3
9. Sinon si $k = 0$ poser $r2 = m$
10. Sinon poser $r2 = s - 1$
11. FinSi
12. Si $r2 = r$
13. Alors incrémenter count
14. Poser $g_i = \text{pgcd}(d_i, b_{m-i})$ pour i de 1 à r
15. Sinon si $r2 > r$ alors poser $r = r2$;
16. Poser $count = 1$;
17. Poser $d_i = b_{m-1}$ pour i de 1 à r
18. FinSi
19. FinTantQue
20. Si $d_r = d_1 \prod_{i=2}^{r} \frac{d_i}{d_{i-1}}$
21. Alors renvoyer r et $(d_1, \ldots, d_r)$
22. Sinon retourner à 2
23. FinSi

2.2 Fonctionnement et mise en œuvre

Cet algorithme se base sur une nouvelle caractérisation des pgcd des mineurs d'une matrice :

Proposition 1 *On garde les notations de l'algorithme.*

On suppose que g est le polynôme caractéristique de B, c'est-à-dire que l'on n'est pas dans la branche 8.

Alors pour tout $i \in [\![1, s]\!]$, b_{m-i} est un multiple de d_i.

Ainsi, en prenant le pgcd à chaque itération de la boucle TantQue, on obtient les pgcd des mineurs de A avec forte probabilité.

On peut également constater que :
- de nombreux calculs sont effectués modulo des nombres premiers, puis reconstruits par le théorème des restes chinois, ce qui améliore les complexités spatiale et temporelle ;
- l'étape 4 peut être effectuée de manière efficace grâce aux structures Toeplitz des matrices ;
- à l'étape 5, on calcule d'abord le polynôme minimal h de B grâce à l'algorithme de Berlekamp-Massey (voir annexe B), puis on en déduit $g = hx^{m-s}$, qui est le polynôme caractéristique de B avec forte probabilité.

2.3 Complexité temporelle

L'auteur souligne que la convergence de cet algorithme n'est pas clairement montrée. En effet, il subsiste un problème lorsque les coefficients cherchés sont de petits nombres premiers.

L'auteur ne le mentionne pas, mais je constate de plus que l'algorithme contient deux retours en arrière, ce qui peut grandement augmenter la complexité de l'algorithme (il peut même ne plus converger du tout si A n'est pas assez générique).

Cependant, lorsque l'algorithme converge, il nécessite $O^{\sim}\left(m^2 \log(\|A\|) \log\left(\frac{1}{\epsilon}\right)\right)$ produits matrice-vecteur (modulo un entier premier) et $O^{\sim}\left(\left(m^2 n \log(\|A\|) + m^3 \log^2(\|A\|)\right) \log\left(\frac{1}{\epsilon}\right)\right)$ opérations sur des bits. Il a une meilleure complexité temporelle que le meilleur algorithme déterministe alors connu.

3 Deuxième algorithme

3.1 Intérêt

L'auteur présente une variante du premier algorithme ayant la même complexité, mais qui converge plus rapidement. Pour cela, on ne se place plus dans $\mathbb{Z}$ mais dans un anneau plus grand $\mathcal{R}$: à l'étape 3, on choisit les coefficients de U, L et D dans $\mathcal{R}$. Cela permet d'éviter le problème concernant les petits nombres premiers (voir paragraphe 2.3).

Ainsi, le polynôme caractéristique de B n'est alors plus à coefficients dans $\mathbb{Z}$, mais à coefficients polynomiaux. La proposition 1 est alors modifiée :

Proposition 2 *On garde les notations de l'algorithme.*

On suppose que g est le polynôme caractéristique de B, c'est-à-dire que l'on n'est pas dans la branche 8.

Alors pour tout $i \in [\![1, s]\!]$, d_i est le contenu du polynôme b_{m-i}.

L'auteur propose donc un algorithme Monte Carlo `FindContents` permettant, à partir d'une boîte noire évaluant un uplet de polynômes en un point, de calculer les contenus de ces polynômes (c'est-à-dire les pgcd de leurs coefficients). On peut ainsi reconstruire un algorithme `DeterminantalDivisors2` (voir paragraphe 3.3), qui calcule les pgcd des mineurs de A, et en déduire un algorithme calculant la forme normale de Smith avec même complexité.

3.2 Brève description de l'algorithme

L'algorithme `FindContents` procède de manière similaire aux étapes 12 à 18 de l'algorithme `DeterminantalDivisors`, notamment en utilisant à nouveau la méthode des pgcd.

3.3 Reconstruction de DeterminantalDivisors2 avec FindContents

On peut reconstruire `DeterminantalDivisors2` :

1. Poser r le rang de A
2. Choisir au hasard des matrices U Toeplitz supérieure, L Toeplitz inférieure et D diagonale, à coefficients dans $\mathcal{R}$
3. Poser $B = UALD$ et définir une boîte noire qui évalue les r coefficients de plus grand ordre du polynôme caractéristique de B en $3n - 2$ points
4. Appliquer `FindContents` en utilisant cette boîte noire

Cet algorithme a des complexités telles `DeterminantalDivisors2` aura les mêmes complexités que `DeterminantalDivisors`.

3.4 Remarques

Je constate deux points qui ne sont pas précisés par l'auteur :
– cet algorithme nécessite de connaître *a priori* le rang de A, et n'est donc pas aussi puissant que `DeterminantalDivisors` ;
– l'auteur ne précise pas du tout comment obtenir la boîte noire de l'étape 3, ce qui ne me paraît pas évident.

4 Remarques

J'ai déjà souligné certains points aux paragraphes 2.3 et 3.4, qui laissent à penser que l'auteur n'a pas détaillé tous les problèmes concernant la mise en œuvre pratique des algorithmes : comment connaître le rang de A, comment gérer les retours en arrière ? De manière générale, les démonstrations concernant la complexité des algorithmes sont parfois confuses, et les problèmes liés à la convergence des algorithmes ne sont pas explicités.

Cependant, cet article est plutôt précurseur en ce qui concerne l'utilisation d'une "boîte noire" et non de l'objet en lui-même, et il a servi de base à d'autres algorithmes utilisant le même principe mais plus fiables [3].

Conclusion Cet article propose un premier algorithme pour le calcul de la forme normale de Smith et, après en avoir indiqué le principal défaut, décrit une amélioration possible. Bien qu'hésitant, il ouvre beaucoup de portes, aussi bien en ce qui concerne les caractérisations nouvelles des mineurs d'une matrice qu'il utilise, que le concept de boîte noire.

Références

[1] M. Trolin : Lattices with many cycles are dense. *Electronic Colloquium on Computational Complexity*, 2004.

[2] M. Giesbrecht : Fast omputation of the smith form of an integer atrix. *Proceedings of ISSAC'95*, pages 110–118, 1995.

[3] A. Storjohann : Computing hermite and smith normal forms of triangular integer matrices. *Linear Algebra and it Applications*, pages 25–45, 1998.

[4] M. Giesbrecht : Probabilistic computation of the smith normal form of a sparse integer matrix. *H. Cohen (Ed.), Algorithmic Number Theory : Second International Symposium*, pages 173–186, 1996.

[5] Berlekamp-massey algorithm. `http://en.wikipedia.org/wiki/Berlekamp-Massey_algorithm`.

[6] N. Ben Atti, G. Diaz-Toca et H. Lombardi : The berlekamp-massey algorithm revisited. pages 25–45, 2005.

A Forme normale de Smith

Toute matrice entière peut se mettre sous une unique forme normale de Smith, définie comme suit :

Théorème 1 *Soit A une matrice à coefficients entiers de rang r.*

Il existe un unique $(s_1, \ldots, s_r) \in \mathbb{Z}^r$ tel que pour tout $i \in [\![1, r-1]\!]$, $s_i | s_{i+1}$ et il existe deux matrices entières unimodulaires P et Q telles que :

$$PAQ = diag(s_1, \ldots, s_r, 0, \ldots, 0)$$

On note d_k le pgcd de tous les mineurs d'ordre k de A. On pose $d_0 = 1$. On a alors la propriété suivante :

Proposition 3 *On a :*

$$\forall i \in [\![1, r-1]\!],\ s_i = d_i / d_{i-1}$$

B Algorithme de Berlekamp-Massey

L'algorithme de Berlekamp-Massey permet de trouver la plus petite relation linéaire entre des nombres, et par extension le polynôme minimal d'une matrice [5]. Il procède par divisions successives de polynômes [6].