Études

Les polynômes d'Ehrhart

Philippe CLAUSS

(suite du n° 418)

V.- Polynômes d'Ehrhart et Informatique Parallèle

Nous nous intéressons plus particulièrement au domaine de la parallélisation automatique de programmes séquentiels en informatique parallèle. Son objectif est de déterminer des méthodes efficaces permettant de détecter et d'exploiter le parallélisme inhérent à une séquence d'opérations d'un programme séquentiel «classique». Afin d'introduire l'idée générale de ces méthodes, prenons un exemple évident d'une suite d'additions :

```
X := A + B ;
Y := C + D ;
Z := X + Y ;
```

Sur un ordinateur possédant un seul processeur, l'ordre de calcul sera d'abord le calcul de X, puis le calcul de Y, et finalement le calcul de Z. Mais pour accélérer le calcul, on peut remarquer que le calcul de Y peut s'effectuer indépendamment du calcul de X. Par conséquent, sur un ordinateur possédant au moins deux processeurs, X et Y peuvent être calculés simultanément sur deux processeurs différents. Par contre, le calcul de Z ne peut évidemment pas s'effectuer avant ceux de X ou de Y. Par conséquent, Z doit être calculé après X et Y sur n'importe quel processeur.

La parallélisation automatique possède son plus large champ d'applications dans les programmes séquentiels de types scientifiques : calculs numériques, simulations physiques… Ces programmes, écrits le plus souvent dans le langage de programmation *Fortran*, ont un parallélisme inhérent situé à 80 % dans des structures de contrôle appelées communément *boucles*. Ces boucles permettent d'exprimer la répétition d'une suite d'opérations relativement à une variation de valeurs d'indices. Elles peuvent être schématisées de

la manière suivante :

```
Pour i de inf(i) a sup(i) faire
    Pour j de inf(j) a sup(j) faire
        ..............................
            operation 1 ;
            operation 2 ;
            ..........
            operation n ;
        ..............................
    Fin Pour
Fin Pour
```

Ainsi, l'indice de la boucle la plus interne, c'est à dire la plus proche de la suite d'opérations est incrémenté de 1 après l'exécution de l'operation n, puis la suite d'opérations est ré-éxécutée. Sa valeur initiale est donnée par la valeur de inf(..) et sa valeur maximum par sup(..). Dès que sa valeur maximum est atteinte, la suite d'opérations est exécutée une dernière fois puis l'indice de la boucle de niveau juste supérieur est incrémenté. Si sa valeur maximum n'est pas atteinte, la boucle la plus interne est à nouveau exécutée pour toutes les valeurs de son indice. Par exemple, un programme permettant d'afficher tous les éléments d'une matrice A de taille N x N pourrait être le suivant :

```
Pour i de 1 a N faire
    Pour j de 1 a N faire
        Afficher[A(i,j)]
    Fin Pour
Fin Pour
```

Un ensemble de structures de boucle de ce type, incluses les unes dans les autres, est appelé communément un *nid de boucles*.

De plus, les bornes d'indices inf(..) et sup(..) sont en général des fonctions affines des indices des boucles de niveaux supérieurs, et les opérations portent souvent sur des éléments de tableaux, ou de matrices, dont les indices sont eux-mêmes exprimés par des fonctions affines des indices de boucles.

Un exemple simple et bien connu est le produit matriciel. Tout mathématicien connaît la formule $c_{ij} = \sum_{k=1}^{N} a_{ik}b_{kj}$, exprimant le produit de deux matrices carrées A et B de tailles N x N en une matrice C. Ce calcul peut être programmé en un nid de boucles :

```
Pour i de 1 a N faire
```

```
    Pour j de 1 a N faire
        Pour k de 1 a N faire
            c[i,j] := c[i,j] + a[i,k] * b[k,j]
        Fin Pour
    Fin Pour
Fin Pour
```

où le calcul d'un élément $c(i,j)$ est effectué en N pas par cumul dans $c(i,j)$.

Nos méthodes de parallélisation automatique passent par une modélisation géométrique d'un tel nid de boucles. Chaque pas de calcul de tous les éléments $c(i,j)$ est représenté dans l'espace par un point de coordonnées entières (i,j,k). Ainsi, le produit matriciel est représenté par un cube défini par les bornes inférieures et supérieures de tous les indices de boucles i, j et k, dans lequel seuls les points à coordonnées entières sont significatifs : ce sont les solutions entières du système défini par :

$$\begin{cases} 1 \leq i \leq N \\ 1 \leq j \leq N \\ 1 \leq k \leq N \end{cases}$$

Dans ce cas, leur nombre, ou le polynôme d'Ehrhart du cube, est facile à déterminer, mais permet de verifier la méthode : $pe(n) = N^3$. Ce nombre nous permet dans ce cas d'exprimer le nombre de calculs élémentaires effectués par notre programme. Cette information est utile notamment pour évaluer le temps d'exécution du programme, ou pour réaliser l'équilibrage de charge entre les processeurs, c'est à dire allouer les calculs élémentaires en nombre égal pour chaque processeur de l'ordinateur.

Voyons maintenant quelques applications particulières des polynômes d'Ehrhart à l'analyse et à la transformation de programmes parallèles.

Le parallélisme potentiel

La parallélisation d'un programme consiste non seulement en une allocation des calculs élémentaires à des processeurs mais également en un ordonnancement dans le temps de ces calculs. Une méthode aujourd'hui bien connue définit une fonction linéaire d'allocation des calculs de la forme :

$$\mathbb{Z}^n \rightarrow \mathbb{Z}^{n-1}$$

$$alloc : X \mapsto D \,.\, X$$

où X est un point à coordonnées entières associé à un calcul, et D est une matrice de taille $(n\text{-}1) \times n$. Cette fonction revient à projeter le polyèdre convexe défini par les points de calculs, que l'on appelle le *domaine de*

calcul, selon un vecteur $\vec{p}$. La matrice D est donc telle que la matrice

$$\begin{pmatrix} D \\ \vec{p} \end{pmatrix}$$

définit une base de $\mathbb{Z}^n$. Ainsi, *alloc(X)* nous donne les coordonnées du processeur effectuant le calcul X.

L'ordonnancement temporel des calculs s'exprime également par une fonction linéaire, exprimant une modélisation discrète du temps, et considérant que tous les calculs élémentaires prennent une même durée, c'est à dire 1 :

$$\mathbb{Z}^n \rightarrow \mathbb{Z}$$

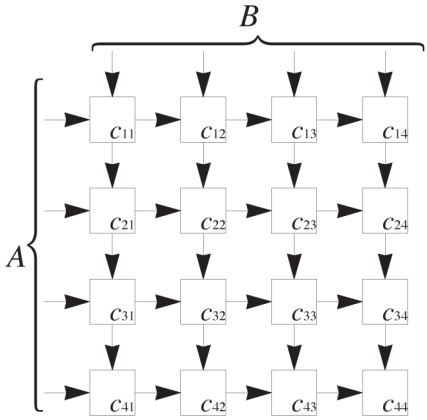
$$pas : \quad X \mapsto \vec{t}.X + c$$

Les vecteurs $\vec{t}$ et $\vec{p}$ doivent vérifier la relation $\vec{p} \cdot \vec{t} \neq 0$, exprimant que deux calculs X et Y de mêmes instants de calculs, c'est à dire tels que $pas(X) = pas(Y)$, ne peuvent pas être effectués par le même processeur. De plus, lorsque le calcul d'un élément utilise un autre élément calculé par le programme, ce calcul ne peut se faire qu'après le calcul de l'élément utilisé. On parle alors de dépendances entre les calculs, modélisées par des *vecteurs de dépendances* $\vec{d}$ entre les points entiers du domaine de calcul. Le respect de ces dépendances est réalisé si et seulement si $\vec{d} \cdot \vec{t} > 0$ pour tous vecteurs $\vec{d}$.

Pour notre exemple du produit matriciel, un ordonnancement temporel et une allocation valides sont définis respectivement par $pas(i,j,k) = i + j + k - 2$ et $alloc(i,j,k) = (i,j)$. Ces choix correspondent à un programme parallèle pouvant être représenté par un réseau carré de taille N x N de processeurs interconnectés et d'exécution rythmée par une horloge globale, où chaque processeur de coordonnées (i,j) est dédié au calcul d'un élément c_{ij} de la matrice C. De plus, les connections entre les processeurs servent à la circulation des éléments des matrices A et B dans un ordre et une direction bien définis, correspondant aux besoins en données de chaque processeur pour son calcul. Un tel réseau est représenté ci-dessous pour $N = 4$.

Les calculs simultanés d'un instant donné t_0, tels que $pas(X) = t_0$, sont par conséquent définis géométriquement par les points entiers appartenant à l'intersection entre un hyperplan orthogonal au vecteur $\vec{t}$ et le domaine de calcul. Leur nombre définit le nombre de calculs simultanés à l'instant t_0, ou parallélisme potentiel à t_0.

Pour notre exemple du produit matriciel avec $pas(i,j,k) = i + j + k - 2$, ces calculs simultanés à t_0 sont définis par :



$$\begin{cases} 1 \leq i \leq N \\ 1 \leq j \leq N \\ 1 \leq k \leq N \\ i + j + k - 2 = t_0 \end{cases}$$

Le nombre de points entiers du polytope paramétrique P_{N,t_0} ainsi défini correspond donc au parallélisme potentiel de l'instant t_0. Le calcul du polynôme d'Ehrhart de P_{N,t_0} donne le résultat suivant :

Si $1 \leq t_0 \leq N$, $\quad pe\,(N, t_0) = \frac{1}{2}\, t_0^2 + \frac{1}{2}\, t_0$

Si $N \leq t_0 \leq 2N - 1$, $\quad pe\,(N, t_0) = -\frac{3}{2}\, N^2 - t_0^2 + 3Nt_0 - t_0$

Si $2N - 1 \leq t_0 \leq 3N - 2$, $\quad pe\,(N, t_0) = -\frac{9}{2}\, N^2 + \frac{1}{2}\, t_0^2 - 3Nt_0 - \frac{3}{2}\, N + \frac{1}{2}\, t_0$

Il est intéressant de maximiser ces polynômes, afin d'obtenir le nombre maximum symbolique de calculs simultanés, *maxpar*(*N*), atteint lors de l'exécution du programme parallèle. Cette information peut être nécessaire pour déterminer le nombre maximum de processeurs simultanément actifs, et donc le nombre minimum de processeurs utiles à l'exécution de ce programme. Cette maximisation ne posant pas de problème mathématique particulier, nous en donnons juste le résultat final :

Si $N \leq 7$, $\quad maxpar(N) = \frac{N\,(N+1)}{2}$

Si $N > 7$, $\quad maxpar(N) = \frac{3}{4}\, N^2 - \frac{3}{2}\, N + \left[\frac{1}{4}, 0\right]$

Ces quantités peuvent être comparées au nombre de processeurs, *np*,

résultant de la fonction d'allocation *alloc*, afin de déterminer le taux d'utilisation du réseau de processeurs. Dans notre exemple, *np* est facile à déterminer ($np(N) = N^2$). Mais dans le cas plus général d'une fonction *alloc* quelconque, la détermination de *np* nécessite l'utilisation de la méthode présentée au paragraphe 4.

Pour l'exemple, le calcul du ratio $\frac{maxpar(N)}{np(N)}$ nous informe qu'environ 3/4 des processeurs au maximum sont utilisés simultanément pour N grand.

La taille des mémoires caches

Nous utilisons ici l'extension des polynômes d'Ehrhart aux projections affines de polytopes décrite au paragraphe 4.

Considérons le programme suivant :

```
Pour i de 1 a 8 faire
    Pour j de 1 a 5 faire
        a(6i+9j-7) := a(6i+9j-7) + 5
    Fin Pour
Fin Pour
```

Un objectif classique en analyse de programmes est de déterminer le nombre de données différentes accédées lors de l'exécution. Ce nombre sert notamment à évaluer la taille mémoire nécessaire au stockage des données utiles au programme. Si celui-ci est exécuté sur un seul processeur d'une machine parallèle à mémoire distribuée, c'est-à-dire où chaque processeur possède une mémoire locale (ou cache), cette taille concernera uniquement la mémoire de ce processeur.

Dans notre exemple, les données accédées sont des éléments d'un tableau à une dimension *a*. Ces éléments sont référencés par une combinaison affine des indices de boucle du programme, $a(6i + 9j - 7)$. Déterminer le nombre de données accédées par le programme revient à déterminer le nombre de valeurs différentes de $6i + 9j - 7$ atteintes pour $1 \leq i \leq 8$ et $1 \leq j \leq 5$. Ce problème revient donc à dénombrer les points entiers résultant de l'application de la projection affine définie par $Proj\_Aff(i,j) = 6i + 9j - 7$, aux points entiers du polytope défini par $P = \{(i,j) \in \mathbb{Z}^2 \mid 1 \leq i \leq 8, 1 \leq j \leq 5\}$.

Ce problème relève bien de la méthode du paragraphe 4. Le polytope paramétré, caractérisant les points de P qui résultent en une même valeur $y = 6i + 9j - 7$, est défini par $Py = \{(i,j) \in \mathbb{Z}^2 \mid 1 \leq i \leq 8, 1 \leq j \leq 5, y = 6i + 9j - 7\}$. On calcule les polynômes d'Ehrhart de P_y. Le résultat consiste en 3 domaines adjacents D_1, D_2 et D_3 de valeurs de y, chacun étant associé à un polynôme d'Ehrhart $pe_1(y)$, $pe_2(y)$ et $pe_3(y)$:

- $D_1 = \{y \in \mathbb{Z} \mid 8 \leq y \leq 44\}$,
  $pe_1(y) = [0, \frac{1}{18}, 0]\, y + [0, -\frac{1}{9}, 0, 0, -\frac{1}{18}, 0, 0, \frac{5}{9}, 0, 0, -\frac{11}{18}, 0, 0, \frac{2}{9}, 0, 0, \frac{1}{18}, 0]$
- $D_2 = \{y \in \mathbb{Z} \mid 44 \leq y \leq 50\}$,
  $pe_2(y) = [0, 3, 0, 0, 2, 0]$
- $D_3 = \{y \in \mathbb{Z} \mid 50 \leq y \leq 86\}$,
  $pe_3(y) = [0, -\frac{1}{18}, 0]\, y + [0, \frac{46}{9}, 0, 0, \frac{95}{18}, 0, 0, \frac{49}{9}, 0, 0, \frac{83}{18}, 0, 0, \frac{52}{9}, 0, 0, \frac{89}{18}, 0]$

Par exemple, l'élément de tableau $a(41)$ est référencé 2 fois par le programme : puisque 41 *modulo 3* = 2 (la 2e valeur du 1er coefficient périodique est prise en compte) et 41 *modulo 18* = 5 (la 5e valeur du 2e coefficient périodique est prise en compte), on obtient $pe_1(41) = \frac{41}{18} - \frac{5}{18} = 2$.

Les domaines de valeurs de y sont transformés en domaines disjoints : $D_1 = \{y \in \mathbb{Z} \mid 8 \leq y \leq 44\}$, $D_2 = \{y \in \mathbb{Z} \mid 44 \leq y \leq 50\}$ et $D_3 = \{y \in \mathbb{Z} \mid 50 \leq y \leq 86\}$. Sur chaque domaine D_q et pour tout polynôme $pe_v(y)$ extrait de $pe_q(y)$, tel qu'il n'est pas égal à zéro pour toute valeur de y, on définit un polytope pour lequel les points entiers doivent être dénombrés. On obtient les polytopes suivants :

$$D_1 \begin{cases} \{8 \leq y \leq 44,\ y \textit{ modulo } 18 = 2\} \\ \{8 \leq y \leq 44,\ y \textit{ modulo } 18 = 5\} \\ \{8 \leq y \leq 44,\ y \textit{ modulo } 18 = 8\} \\ \{8 \leq y \leq 44,\ y \textit{ modulo } 18 = 11\} \\ \{8 \leq y \leq 44,\ y \textit{ modulo } 18 = 14\} \\ \{8 \leq y \leq 44,\ y \textit{ modulo } 18 = 17\} \end{cases}$$

$$D_2 \begin{cases} \{44 \leq y \leq 50,\ y \textit{ modulo } 6 = 2\} \\ \{44 \leq y \leq 50,\ y \textit{ modulo } 6 = 5\} \end{cases}$$

$$D_3 \begin{cases} \{50 \leq y \leq 86,\ y \textit{ modulo } 18 = 2\} \\ \{50 \leq y \leq 86,\ y \textit{ modulo } 18 = 5\} \\ \{50 \leq y \leq 86,\ y \textit{ modulo } 18 = 8\} \\ \{50 \leq y \leq 86,\ y \textit{ modulo } 18 = 11\} \\ \{50 \leq y \leq 86,\ y \textit{ modulo } 18 = 14\} \\ \{50 \leq y \leq 86,\ y \textit{ modulo } 18 = 17\} \end{cases}$$

L'application de la transformation décrite dans la proposition 4 donne les polytopes suivants :

$$D_1 \begin{cases} \{1 \le z \le 2\} \\ \{1 \le z \le 2\} \\ \{0 \le z \le 2\} \\ \{0 \le z \le 1\} \\ \{0 \le z \le 1\} \\ \{0 \le z \le 1\} \end{cases} \qquad D_2 \begin{cases} \{7 < z < 8\} \\ \{7 \le z \le 7\} \end{cases}$$

$$D_3 \begin{cases} \{3 \le z \le 4\} \\ \{3 \le z \le 4\} \\ \{3 \le z \le 4\} \\ \{3 \le z \le 4\} \\ \{2 \le z \le 4\} \\ \{2 \le z \le 3\} \end{cases}$$

Le dénombrement des points entiers de ces polytopes est facile et résulte en $2 + 2 + 3 + 2 + 2 + 2 + 0 + 1 + 2 + 2 + 2 + 2 + 3 + 2 = 27$.

Pour tous les polynômes $pe_v(y)$ non toujours nuls, on calcule leurs racines afin de déterminer N_r. Par exemple, on a les polynômes $pe_v(y)$ suivants sur le domaine D_1 :

$$D_1 \begin{cases} \textit{Si } y \textit{ modulo } 18 = 2, & pe_2(y) = \frac{1}{18}y - \frac{1}{9} \\ \textit{Si } y \textit{ modulo } 18 = 5, & pe_5(y) = \frac{1}{18}y - \frac{5}{18} \\ \textit{Si } y \textit{ modulo } 18 = 8, & pe_8(y) = \frac{1}{18}y + \frac{5}{9} \\ \textit{Si } y \textit{ modulo } 18 = 11, & pe_{11}(y) = \frac{1}{18}y - \frac{11}{18} \\ \textit{Si } y \textit{ modulo } 18 = 14, & pe_{14}(y) = \frac{1}{18}y - \frac{2}{9} \\ \textit{Si } y \textit{ modulo } 18 = 17, & pe_{17}(y) = \frac{1}{18}y + \frac{1}{18} \end{cases}$$

Le calcul de leurs racines résulte en une seule racine valide 11. Pour les autres domaines, on détermine une seule autre racine valide sur D_3 qui est 83. Par conséquent, l'on doit soustraire 2 au décompte précédent.

En conclusion, le nombre de données différentes accédées par le programme, ou la taille mémoire nécessaire au programme, est de 25.

VI.- Conclusion

Les extensions et applications des polynômes d'Ehrhart nous ont amenés à des publications dans des conférences et revues d'audience internationales spécialisées en informatique [1, 2, 3, 4], et bien d'autres sont à prévoir. Ces

résultats sont aujourd'hui reconnus et utilisés dans d'autres travaux [14, 15, 18, 16]. Beaucoup d'autres utilisations auraient pu être présentées ici, et peuvent être espérées moyennant une modélisation opportune des problèmes posés. L'analyse symbolique permise par les travaux d'Eugène Ehrhart, est non seulement intéressante pour résoudre des problèmes de nature paramétrée, mais l'est aussi pour tout autre type de problèmes gr\^ace à des modélisations constructives. Pour les lecteurs ayant accès à internet, une page régulièrement mise à jour, et donnant toutes les informations et références de nos travaux sur le sujet est accessible à l'adresse suivante : http://icps.u-strasbg.fr/Ehrhart/Ehrhart.html

Les polynômes d'Ehrhart n'intéressent pas uniquement l'informatique. Les mathématiques s'y intéressent également beaucoup aujourd'hui, se posant des questions telles que le sens géométrique des coefficients de ces polynômes.

Rappelons qu'E. Ehrhart a obtenu deux prix de l'Académie des Sciences pour ses travaux arithmo-géométriques.

Remerciements

Je tiens à remercier Monsieur Eugène Ehrhart pour l'honneur qu'il me fait de nos rencontres, de son aide et son soutien à mes travaux.

Bibliographie

[1] Ph. Clauss. Counting Solutions to Linear and Nonlinear Constraints through Ehrhart polynomials: Applications to Analyze and Transform Scientific Programs. *10th ACM Int. Conf. on Supercomputing*, Philadelphia, May 1996. Egalement accessible comme rapport de recherches ICPS 96-03. http://icps.u-strasbg.fr/pub-96/pub-96-03.ps.gz

[2] Ph. Clauss et V. Loechner. Parametric Analysis of Polyhedral Iteration Spaces. *IEEE Int. Conf. on Application Specific Array Processors, ASAP'96*, Chicago, Illinois, August 1996. Version étendue à paraitre dans *Journal of VLSI Signal Processing*, Kluwer Academic Publishers, Boston, USA, 1997. Egalement accessible comme rapport de recherches ICPS 96-04. http://icps.u-strasbg.fr/pub-96/pub-96-04.ps.gz

[3] Ph. Clauss. Handling Memory Cache Policy with Integer Points Countings. *Third International Conference Euro-par'97*, Passau, Allemagne, Ao˚t 1997. Lecture Notes in Computer Science 1300, Springer, pages 285-293. Egalement accessible comme rapport de recherches ICPS 97-02. http://icps.u-strasbg.fr/pub-97/pub-97-02.ps.gz

[4] Ph. Clauss, V. Loechner et D. Wilde. Deriving Formulae to Count Solutions to Parameterized Linear Systems using Ehrhart Polynomials: Applications to the

Analysis of Nested-Loop Programs. Soumis à *Journal of Symbolic Computation*. Egalement accessible comme rapport de recherches ICPS 97-05. http://icps.u-strasbg.fr/pub-97/pub-97-05.ps.gz

[5] E. Ehrhart. Sur les polyèdres rationnels homothétiques à n dimensions. *C.R. Acad. Sci.* Paris, 254:616-618, 1962.

[6] E. Ehrhart. Sur un problème de géométrie diophantienne linéaire I. *J. Reine Angew. Math.*, 226:1-29, 1967.

[7] E. Ehrhart. Sur un problème de géométrie diophantienne linéaire II. *J. Reine Angew. Math.*, 227:25-49, 1967.

[8] E. Ehrhart. Une méthode géométrique en combinatoire, *L'Ouvert*, bulletin de la régionale alsacienne de l'APMEP, déc. 1976.

[9] E. Ehrhart. Sur les systèmes d'inéquations diophantiennes linéaires, *J. Reine Angew. Math.*, 262/263, 45-57, 1973.

[10] E. Ehrhart. *Polynômes arithmétiques et Méthode des Polyèdres en Combinatoire*. International Series of Numerical Mathematics, vol.35, Birkhäuser Verlag, Basel/Stuttgart, 1977.

[11] E. Ehrhart. Sur les polygones entiers ou rationnels, *L'Ouvert*, bulletin de la régionale alsacienne de l'APMEP, sept. 1993.

[12] E. Ehrhart. Un théorème arithmo-géométrique et ses généralisations, *L'Ouvert*, bulletin de la régionale alsacienne de l'APMEP, déc. 1994.

[13] E. Ehrhart. La loi de réciprocité diophantienne, *L'Ouvert*, bulletin de la régionale alsacienne de l'APMEP, n. 394, 1994.

[14] S. Ghosh, M. Martonosi et S. Malik. Cache Miss Equations: An Analytical Representation of Cache Misses. *11th ACM Int. Conf. on Supercomputing*, Vienne, Autriche, juillet 1997.

[15] D.-C. R. Ju, J.-F. Collard and K. Oukbir: Probabilistic Memory Disambiguation and its Application to Data Speculation. Rapport de recherche PRiSM 96-010, Université de Versailles, 1996. ftp://ftp.prism.uvsq.fr/pub/reports/1996/1996.010.ps.gz

[16] V. Loechner. Contribution à l'étude des polyèdres paramétrés et applications en parallélisation automatique. Thèse d'informatique de l'Université Louis Pasteur, décembre 1997.

[17] V. Loechner and D. K. Wilde. Parameterized polyhedra and their vertices. *International Journal of Parallel Programming*, vol. 25, no. 6, décembre 1997. Egalement rapport de recherche ICPS 96-09, Université Louis Pasteur, 1996. http://icps.u-strasbg.fr/pub-96/pub-96-09.ps.gz

[18] R. Sakellariou, J. R. Gurd. Compile-Time Minimisation of Load Imbalance in Loop Nests. *11th ACM Int. Conf. on Supercomputing*, Vienne, Autriche, juillet 1997.

[19] G. M. Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics, Springer-Verlag, New-York, 1995.