4. Nouvelle décomposition de matrices pour le calcul *in situ*

La décomposition d'une application linéaire F en série d'affectations pour en tirer un programme *in situ* est toujours possible en $2n-1$ étapes, ce qui est très peu.

La lecture de ces $2n-1$ affectations (toutes linéaires aussi) s'interprète comme la décomposition de la matrice de départ en un produit de matrices simples : chacune est obtenue en modifiant une unique ligne à la matrice identité. C'est là un nouveau type de décomposition d'une matrice en produit de matrices simples.

Donnons un exemple avec un calcul modulo 2.

Nous voulons aboutir à l'affectation finale :

$F(A, B, C, D) = (A + D, A + B + C, A + C + D, A + C)$.

Comme toute application linéaire, on peut l'écrire sous la forme du produit d'une matrice et d'un vecteur colonne.

Multiplication de matrices

$$F = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix} = \begin{pmatrix} A+D \\ A+B+C \\ A+C+D \\ A+C \end{pmatrix}$$

L'application est définie par le vecteur colonne indiqué. Toutefois, le calcul ne peut se faire *in situ* d'un seul coup car, dès que l'on a remplacé A par $A + D$, il faut réaménager le calcul en fonction de cette nouvelle valeur de A, et ainsi de suite pour toutes les variables. Cela peut se faire en utilisant des pseudomatrices unités où une seule ligne est transformée. Ainsi, la matrice F de la transformation est décomposée en un produit des matrices J, K, L et M, explicitées ci-dessus. On obtient avec ces matrices :

$F(A, B, C, D) = MLKJ\,(A, B, C, D) = (A + D, A + B + C, A + C + D, A + C)$.

Multiplication de matrices

$$J = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix} = \begin{pmatrix} A+D \\ B \\ C \\ D \end{pmatrix}$$

Multiplication de matrices — Calcul modulo 2

$$K = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} A+D \\ B \\ C \\ D \end{pmatrix} = \begin{pmatrix} A+D \\ A+B+C+2D \\ C \\ D \end{pmatrix} = \begin{pmatrix} A+D \\ A+B+C \\ C \\ D \end{pmatrix}$$

Multiplication de matrices

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} A+D \\ A+B+C \\ C \\ D \end{pmatrix} = \begin{pmatrix} A+D \\ A+B+C \\ A+C+D \\ D \end{pmatrix}$$

Multiplication de matrices — Calcul modulo 2

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} A+D \\ A+B+C \\ A+C+D \\ D \end{pmatrix} = \begin{pmatrix} A+D \\ A+B+C \\ A+C+D \\ A+C+2D \end{pmatrix} = \begin{pmatrix} A+D \\ A+B+C \\ A+C+D \\ A+C \end{pmatrix}$$

Puisqu'il faut $n + 1$ affectations pour une permutation circulaire entre n variables, il faut dans le pire cas (quand tous les cycles sont des cycles d'ordre 2) $3n/2$ affectations si n est pair, et $(3n - 1)/2$ affectations si n est impair. Dans le cas d'une permutation se décomposant en k cycles non réduits à un point, $c_1, c_2, \ldots, c_k$ de longueurs $|c_1|, |c_2|, \ldots, |c_k|$, le programme *in situ*, résultat de l'idée indiquée au-dessus, est composé de $k + |c_1| + |c_2| + \ldots + |c_k|$ affectations. Ce nombre est le meilleur qu'on puisse obtenir. La décomposition d'une permutation quelconque de variables en un programme *in situ* de $3n/2$ opérations au plus permet d'échanger de l'information entre n fils qui ne peuvent pas se croiser (car placés sur un même plan), en utilisant $3n/2$ portes logiques *xor* au plus *(voir l'encadré 3)*.

D'autres résultats ont été obtenus pour les applications linéaires. Tout d'abord sur des variables booléennes et plus généralement modulo p avec p premier (S. Burckel et Marianne Morillon) et par la suite sur des entiers modulo k, pour tout entier k fixé (Emmanuel Thomé). Un exemple est donné dans l'encadré ci-dessus.

Les résultats obtenus indiquent que toute transformation linéaire peut s'obtenir par un programme *in situ* comportant au plus $2n-1$ affectations, chacune correspondant à une opération elle-même linéaire (comme dans l'exemple). De plus, on peut imposer l'ordre dans lequel se feront les affectations : 1, 2, ..., n, $n-1$, ..., 2, 1 (affectation de la première variable, de la deuxième, etc.)

Trouver le simple est compliqué

Toute transformation n'est pas linéaire, aussi fallait-il étudier le cas général. Dans un premier temps, il a été établi que toutes les transformations bijectives (linéaires ou non) F de S^n dans S^n possédaient des programmes *in situ* et que $2n - 1$ affectations suffisent. S. Burckel et Emeric Gioan ont ensuite montré que $5n-4$ affectations suffisent toujours, quelle que soit la complexité de F et cela même quand F n'est pas bijective. Lorsque le nombre d'éléments de S est une puissance de 2, on peut même descendre jusqu'à $4n-3$.

Ces résultats récents améliorent un résultat précédent obtenu par S. Burckel et M. Morillon qui indiquait que n^2 affectations suffisaient dans le cas booléen. Le gain de n^2 à $4n - 3$ est considérable. Malheureusement, le calcul des décompositions est parfois très difficile et lorsque le nombre de variables est grand, on risque de se heurter à une impossibilité pratique. Autrement dit : la complexité des algorithmes pour calculer les décompositions les plus simples est grande. On a là, une fois encore, une illustration de la maxime « Trouver le simple est compliqué ».

Ces résultats ne donnent pas nécessairement les décompositions optimales, qui sont encore plus difficiles à trouver. Les explorations exhaustives sont en théorie envisageables, puisque nous sommes dans un cas fini. Elles garantissent de trouver les programmes *in situ* optimaux, mais elles sont si coûteuses en calcul qu'on ne peut les envisager que si n est vraiment petit (inférieur à 10). De nombreux résultats sont donc encore attendus dans ce domaine à la fois combinatoire, algébrique et algorithmique. Il est à parier qu'à mesure des perfectionnements apportés aux méthodes pratiques pour calculer les décompositions en programmes *in situ*, leur intérêt concret apparaîtra et leur utilisation se généralisera.