Université Paris Descartes
UFR de Mathématiques et Informatique
45, rue des Saints-Pères 75270 Paris cedex 06

MASTER MA-IM 2012–2013

Logiciels mathématiques

Projet

À remettre pour le 13 décembre 2012 :
le fichier des fonctions Scilab, le fichier TeX et le fichier compilé au format PDF
ne pas oublier vos noms dans les fichiers source ".sce", ".sci" et ".tex"

Description du problème :
On s'intéresse au nombre d'opérations scalaires à effectuer pour calculer un produit de n matrices réelles : $A_1 A_2 \cdots A_n$.
Notations : La matrice A_i a comme dimensions $[d_i, d_{i+1}]$ $(1 \leq i \leq n)$ et on note $d = (d_1, d_2, \ldots, d_{n+1})$.
Par ailleurs, on note $C(i,j)$ le nombre minimal d'opérations scalaires nécessaires pour calculer le produit $A_i \cdots A_j$ ($1 \leq i < j \leq n$).

Rappelons d'abord quelques notions de base :

(i) Si A est de taille $[p,q]$ et B de taille $[q,r]$ alors le produit $C = AB$ est défini et on utilise la formule $C_{ij} = \displaystyle\sum_{k=1}^{q} A_{ik} B_{kj}$ pour $1 \leq i \leq p$ et $1 \leq j \leq r$.
La matrice C est de taille $[p,r]$ et le nombre de multiplications scalaires pour la calculer est pqr (pour simplifier on ne tient pas compte des additions).

(ii) La multiplication des matrices est associative, $A_i(A_{i+1}A_{i+2}) = (A_i A_{i+1})A_{i+2}$ $(1 \leq i \leq n-2)$.
Mais le nombre d'opérations scalaires n'est pas le même.

Méthode :
Soit $1 \leq i < j \leq n$ et $i \leq k < j$ tel que $(A_i \cdots A_k)(A_{k+1} \cdots A_j)$ soit le parenthésage optimal, *i.e.* nécessitant le moins d'opérations scalaires, pour évaluer le produit $A_i \cdots A_j$.
Ce choix est optimal si et seulement si chacun des produits $A_i \cdots A_k$ et $A_{k+1} \cdots A_j$ a aussi été calculé de façon optimale. Le problème à résoudre est ainsi décomposé en sous-problèmes plus simples et similaires au problème initial et on a la relation

$$C(i,j) = \min_{i \leq k < j} \Big(C(i,k) + C(k+1,j) + d_i d_{k+1} d_{j+1} \Big), \quad 1 \leq i < j \leq n.$$

Le coût optimal pour le calcul de $A_1 \cdots A_n$ est alors donné par la valeur de $C(1,n)$ obtenue de façon récursive à partir des sous-problèmes. Afin de rendre cette récurrence calculable en un temps raisonnable, on va utiliser une récurrence ascendante pour remplir le tableau de coût C :

on détermine le coût des produits	$A_i A_{i+1}$,	pour $i = 1$ à $n-1$;
ensuite le coût minimal des produits	$A_i A_{i+1} A_{i+2}$,	pour $i = 1$ à $n-2$;
puis le coût minimal de	$A_i A_{i+1} A_{i+2} A_{i+3}$,	pour $i = 1$ à $n-3$;
$\vdots$		
jusqu'à	$A_1 A_2 \cdots A_n$.	

D'où l'algorithme suivant, où $L = j - i + 1$ correspond au nombre de matrices dans le produit $A_i \cdots A_j$.

Algorithme :
Initialiser C et S à la matrice nulle ;
pour $L = 2$ à n
  pour $i = 1$ à $n - (L - 1)$
    $j = i + L - 1$
    $C(i, j) = C(i + 1, j) + d(i)d(i + 1)d(j + 1)$
    $S(i, j) = i$
    pour $k = i + 1$ à $j - 1$
      $c = C(i, k) + C(k + 1, j) + d(i)d(k + 1)d(j + 1)$
      si $c < C(i, j)$ **alors**
        $C(i, j) = c$
        $S(i, j) = k$

Le tableau S va contenir la valeur optimale de k, *i.e.* l'endroit où il faut fermer la parenthèse : si $A_i \cdots A_j = (A_i \cdots A_k)(A_{k+1} \cdots A_j)$ est optimal de coût $C(i, j)$ alors $S(i, j) = k$ $(i \leq k < j)$.

Partie calculs et programmation

1. Écrire la fonction Scilab `function [C,S] = tableau_cout(d)` qui détermine les matrices C et S d'après l'algorithme précédent en utilisant comme unique entrée le vecteur des dimensions d.

2. Écrire la fonction Scilab `function demonstration(d)` qui a comme entrée le vecteur d et qui présente, sous une forme de votre choix, la façon optimale de parenthésage de $A_1 \cdots A_n$.
   Exemple : si $d = (5, 4, 6, 2, 7)$, on trouve $C(1, 4) = 158$ et un parenthèsage optimal est donné par $(A_1(A_2A_3))A_4$.

Partie rédaction du rapport

Le fichier de rédaction comportera les sections suivantes :

1. Une introduction, avec la description et l'illustration du problème. On vérifiera en particulier les affirmations (i) et (ii).

2. Une étude de la complexité de l'algorithme présenté ici et une comparaison avec la complexité d'une recherche exhaustive du meilleur parenthésage.

3. Explication détaillée de l'algorithme et de son implémentation.

4. Des exemples d'application.

Note 1 : cette liste n'est pas exhaustive, vous avez le droit d'ajouter d'autres points !

Note 2 : vous avez le droit de vous documenter, mais il faut dans ce cas impérativement citer vos sources !