Escritura del problema de parentización óptima

María Camila Aguirre Collante[1] Jessica Tatiana Naizaque Guevara[1]

[1]Departamento de Ingeniería de Sistemas, Pontificia Universidad Javeriana
Bogotá, Colombia
{aguirrec.mcamila, j.naizaque}@javeriana.edu.co

1 de septiembre de 2022

Resumen

En este documento se presenta la formalización del problema de parentización óptima de multiplicaciones escalares de una composición de matrices y la solución dada por el algoritmo *"bottom-up"* con *"backtracking"* que surge a partir de la estrategia de "programación dinámica". **Palabras clave:** parentización, multiplicaciones escalares, cadena matricial, algoritmo, formalización.

Índice

1. Introducción

El producto de una cadena de matrices se puede calcular mediante la agrupación de pares de matrices después de decidir el orden en el cual estas serán multiplicadas. Es importante tener en cuenta que esta operación entre matrices es asociativa, por lo cual, el resultado siempre será el mismo. Sin embargo, es fundamental conocer la mejor manera de asociar dichas matrices, con el fin de reducir la cantidad de multiplicaciones escalares. En el presente documento es posible encontrar un algoritmo que mediante la parentización óptima, minimiza la cantidad de multiplicaciones escalares de una secuencia de matrices y muestra la mejor forma de parentizarlas. A lo largo de este informe se evidencia: la formalización del problema (sección 2), la escritura formal del algoritmo (sección 3) y un análisis experimental del algoritmo mencionado (sección 4).

2. Formalización del problema

Dada una secuencia (cadena) $< A_1, A_2, ..., A_n >$ de n matrices, encontrar la mejor forma de asociar las multiplicaciones matriciales de manera que la cantidad de multiplicaciones escalares se reduzcan. Por ejemplo:

- Dadas las matrices: $A_1 : \mathbb{R}^{10\times100}$, $A_2 : \mathbb{R}^{100\times5}$ y $A_3 : \mathbb{R}^{5\times50}$, se obtiene:

- $((A_1 A_2)\, A_3) = (10 \cdot 100 \cdot 5) + (10 \cdot 5 \cdot 50) = 7500$
- $(A_1\, (A_2 A_3)) = (100 \cdot 5 \cdot 50) + (10 \cdot 100 \cdot 50) = 75000$

donde cada resultado obtenido para cada agrupación representa la cantidad de multiplicaciones escalares; por lo cual, para este ejemplo sería apropiado escoger la primera asociación.

Es importante resaltar que esta cadena de matrices debe cumplir con la condición de que la cantidad de columnas de la matriz A_i sea igual a la cantidad de filas de la matriz A_{i+1}. Para cada una de las matrices presentadas en la cadena inicial sin tener en cuenta A_1 y A_n, donde n determina la cantidad de matrices a tener en cuenta en el proceso. En ese orden de ideas, es posible definir una secuencia $D =< d_i \mid d_i : 0 \leq i \leq n >$, donde, d_0 hace referencia a la cantidad de filas de la matriz A_1, d_n es la cantidad de columnas de la matriz A_n y, en otros casos, $d_i = r_i = c_{i+1}$.

2.1. Definición del problema de "parentización óptima"

Así, el problema de parentización óptima se define a partir de:

1. Una cadena matricial $< A_1, A_2, ..., A_n >$ de n matrices,
2. donde la matriz A_i tiene dimensiones $r_i \times c_i$, $i = 1, 2, ..., n$

calcular la agrupación apropiada que minimice la candidad de multiplicaciones escalares del producto de la dicha cadena.

- Entradas:
  - $D =< d_i \in \mathbb{N} \mid d_i : 0 \leq i \leq n >$, donde, d_0 hace referencia a la cantidad de filas de la matriz A_1, d_n es la cantidad de columnas de la matriz A_n y, en otros casos, $d_i = r_i = c_{i+1}$.
- Salidas:
  - $Z =< A_i; A_i \mathbb{R}^{r_i c_i} \wedge ((r_i \wedge c_i) \in \mathbb{N}) \wedge r_i = c_{i+1} >$

3. Algoritmo de solución: *"bottom-up"* con *"backtracking"*

La idea de este algoritmo es minimizar la cantidad de multiplicaciones escalares en una cadena de matrices, por lo tanto, se requiere solucionar con un proceso que permita optimizarlo. De acuerdo con esto, se sigue una serie de pasos que hacen parte de la estrategia de programación dinámica. Por lo que, para este caso es importante ejecutar los pasos de soluciones calculadas con memoización y bottom-up. Sin embargo, el paso más importante de este proceso es el backtracing. Por lo tanto, en un inicio se recorrerá la matriz con los datos encontrados anteriormente y se irá llenando una tabla de backtracking con el fin de utilizar no solo los índices con la tabla de memoización sino también los valores que van teniendo lugar en este análisis. Así, finalmente, se hará uso de una llamada a un método externo que trabaje recursivamente y que con ayuda del pivote elegido y la estrategia "divide y vencerás" obtendrá de forma recurrente la óptima ubicación de los paréntesis en el proceso de generar el resultado.

4. Análisis experimental

En esta sección se presentarán algunos de los experimentos para confirmar la efectividad del algoritmo presentado en la sección 3. Para realizar la prueba del algoritmo se decidió probar los siguientes casos, con sus respectivos resultados esperados:

1. Secuencia D que contenga las dimensiones de las 3 matrices: $A_1 : \mathbb{R}^{10\times100}$, $A_2 : \mathbb{R}^{100\times5}$ y $A_3 : \mathbb{R}^{5\times50}$. Por lo que $D =< 10, 100, 5, 50 >$ y se espera como resultado la asociación $((A_1 A_2)\, A_3)$.
2. Secuencia D que contenga las dimensiones de las 4 matrices: $A_1 : \mathbb{R}^{5\times7}$, $A_2 : \mathbb{R}^{7\times4}$, $A_3 : \mathbb{R}^{4\times3}$ y $A_4 : \mathbb{R}^{3\times5}$. Por lo que $D =< 5, 7, 4, 3, 5 >$ y se espera como resultado la asociación $((A_1\, (A_2 A_3))\, A_4)$.

Algoritmo 1 backtracking

```
1: procedure MatSeqBT(D)
2:     wT ← D.length − 1
3:     hT ← D.length − 1
4:     let T[1...hT, 1...wT] and B[1...hT − 1, 2...wT] two sequences
5:     for i ← hT − 2 to − 1 do
6:         for j ← i + 1 to wT − 1 do
7:             q ← ∞
8:             m ← 0
9:             for k ← i to j − 1 do
10:                l ← T[i + 1, k + 1]
11:                r ← T[k + 2, j + 1]
12:                if i = 0 then
13:                    x ← D.last
14:                else
15:                    x ← D[i]
16:                end if
17:                v ← l + r + x ∗ D[k + 1] ∗ D[j + 1]
18:                if v < q then
19:                    q ← v
20:                    m ← k + 1
21:                end if
22:            end for
23:            T[i + 1, j + 1] ← q
24:            B[i + 1, j + 1] ← m
25:        end for
26:    end for
27:    show(T)
28:    show(B)
29:    show( Parentization(B, 1, D.length − 1) )
30:    return T[1, wT]
31: end procedure
```

Algoritmo 2 backtracking

```
1: procedure Parentization(B, b, e)
2:     if b = e then
3:         Z ← "A" + b
4:         return Z
5:     else
6:         q ← ⌊(b + e) ÷ 2⌋
7:         Z ← ""
8:         Z ← Z ∗ "("
9:         Z ← Z∗ Parentization(B, b, q)
10:        Z ← Z∗ Parentization(B, q + 1, e)
11:        Z ← Z ∗ ")"
12:        return Z
13:    end if
14: end procedure
```

3. Secuencia D que contenga las dimensiones de las 6 matrices: $A_1 : \mathbb{R}^{5\times7}$, $A_2 : \mathbb{R}^{7\times4}$, $A_3 : \mathbb{R}^{5\times7}$, $A_4 : \mathbb{R}^{7\times4}$, $A_5 : \mathbb{R}^{4\times3}$ y $A_6 : \mathbb{R}^{3\times5}$. Por lo que $D =< 10, 20, 40, 3, 12, 5, 7 >$ y se espera como resultado la asociación $((A_1\,(A_2A_3))\,((A_4A_5)\,A_6))$.

4. Secuencia D que contenga las dimensiones de las 15 matrices: $A_1 : \mathbb{R}^{10\times100}$, $A_2 : \mathbb{R}^{100\times20}$, $A_3 : \mathbb{R}^{20\times3}$, $A_4 : \mathbb{R}^{3\times2}$, $A_5 : \mathbb{R}^{2\times8}$, $A_6 : \mathbb{R}^{8\times40}$, $A_7 : \mathbb{R}^{40\times64}$, $A_8 : \mathbb{R}^{64\times7}$, $A_9 : \mathbb{R}^{7\times5}$, $A_{10} : \mathbb{R}^{5\times11}$, $A_{11} : \mathbb{R}^{11\times50}$, $A_{12} : \mathbb{R}^{50\times14}$, $A_{13} : \mathbb{R}^{14\times15}$, $A_{14} : \mathbb{R}^{15\times20}$ y $A_{15} : \mathbb{R}^{20\times5}$. Por lo que $D =< 10, 100, 20, 3, 2, 8, 40, 64, 7, 5, 11, 50, 14, 15, 20, 5 >$.

5. Secuencia D que contenga las dimensiones de las 18 matrices: $A_1 : \mathbb{R}^{5\times10}$, $A_2 : \mathbb{R}^{10\times3}$, $A_3 : \mathbb{R}^{3\times7}$, $A_4 : \mathbb{R}^{7\times100}$, $A_5 : \mathbb{R}^{100\times8}$, $A_6 : \mathbb{R}^{8\times4}$, $A_7 : \mathbb{R}^{4\times11}$, $A_8 : \mathbb{R}^{11\times20}$, $A_9 : \mathbb{R}^{20\times3}$, $A_{10} : \mathbb{R}^{3\times14}$, $A_{11} : \mathbb{R}^{14\times40}$, $A_{12} : \mathbb{R}^{40\times5}$, $A_{13} : \mathbb{R}^{5\times12}$, $A_{14} : \mathbb{R}^{12\times16}$, $A_{15} : \mathbb{R}^{16\times28}$, $A_{16} : \mathbb{R}^{28\times3}$, $A_{17} : \mathbb{R}^{3\times2}$ y $A_{18} : \mathbb{R}^{2\times1}$. Por lo que $D =< 5, 10, 3, 7, 100, 8, 4, 11, 20, 3, 14, 40, 5, 12, 16, 28, 3, 2, 1 >$.

Luego de ejecutar cada uno de los casos presentados previamente se obtienen los siguientes resultados:

1. Considerando la Figura 1, es posible decir que para este caso, el algoritmo funciona correctamente.

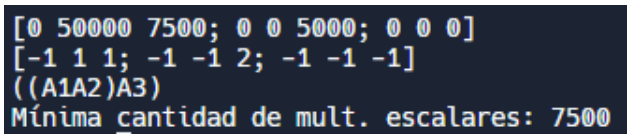

Figura 1: Resultados del primer caso

2. Considerando la Figura 2, es posible decir que para este caso, el algoritmo no funciona completamente, ya que determina de manera correcta la cantidad mínima de multiplicaciones escalares, sin embargo, no muestra la cadena de caracteres correspondiente.

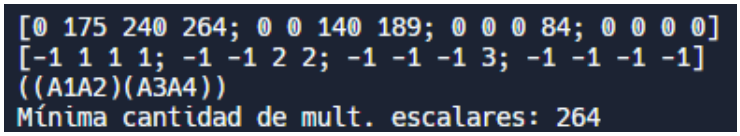

Figura 2: Resultados del segundo caso

3. Considerando la Figura 3, es posible decir que para este caso, el algoritmo no funciona completamente, ya que determina de manera correcta la cantidad mínima de multiplicaciones escalares, sin embargo, no muestra la cadena de caracteres correspondiente. Aunque, cabe resaltar que la cadena generada es bastante cercana a la esperada.

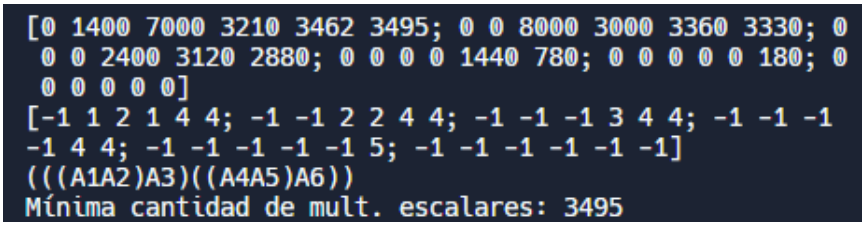

Figura 3: Resultados del tercer caso

4. Considerando la Figura 4 y observando la parentización final, aparentemente se podría afirmar que el algoritmo cumple con su objetivo, sin embargo, como no es posible verificar el resultado esperado y teniendo en cuenta los casos anteriores, es muy probable que el algoritmo no esté trabajando de la mejor forma.

```
[0 5000 15000 9150 6220 6300 7260 12620 12946 12996 13166 14656 15696 1
6126 16776; 0 0 20000 9000 6120 6280 7560 13160 12916 12946 13176 15056
 15736 16176 16876; 0 0 0 6000 4120 5720 12760 22680 12176 11846 13156
22056 16256 16876 18476; 0 0 0 0 120 440 2360 8440 7056 7046 7396 10056
 10016 10476 11276; 0 0 0 0 0 48 880 6144 6698 6756 6902 8236 9420 9846
 10476; 0 0 0 0 0 0 640 5760 6656 6726 6836 7936 9336 9756 10356; 0 0 0
 0 0 0 0 20480 20160 16640 17080 21390 23450 24540 26240; 0 0 0 0 0 0 0
 0 17920 15040 17240 27790 24090 25340 27840; 0 0 0 0 0 0 0 0 0 2240 53
13 20990 12970 14340 17440; 0 0 0 0 0 0 0 0 0 0 385 4235 6740 7825 9500
; 0 0 0 0 0 0 0 0 0 0 0 2750 6250 7300 8800; 0 0 0 0 0 0 0 0 0 0 0 0 77
00 10010 13310; 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10500 18200; 0 0 0 0 0 0 0 0
0 0 0 0 0 0 4200; 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[-1 1 2 1 1 5 5 5 5 5 5 5 5 5 5 5; -1 -1 2 2 2 5 5 5 5 5 5 5 5 5 5; -1 -1
 -1 3 3 5 5 5 5 5 5 5 5 5 5 5; -1 -1 -1 -1 4 5 5 5 5 5 5 5 5 5 5; -1 -1 -
1 -1 -1 5 5 5 5 5 5 5 5 5 5 5; -1 -1 -1 -1 -1 -1 6 7 8 9 10 11 12 13 14;
-1 -1 -1 -1 -1 -1 -1 7 7 7 10 10 10 10 10; -1 -1 -1 -1 -1 -1 -1 -1 8 8
10 10 10 10 10; -1 -1 -1 -1 -1 -1 -1 -1 -1 9 9 10 10 10 10; -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 10 11 10 10 10; -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 11
12 13 14; -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 12 13 14; -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 13 13; -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -
1 14; -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
((((A1A2)(A3A4))((A5A6)(A7A8)))(((A9A10)(A11A12))((A13A14)A15)))
Mínima cantidad de mult. escalares: 16776
```

Figura 4: Resultados del cuarto caso

5. Considerando la Figura 5y observando la parentización final, aparentemente se podría afirmar que el algoritmo cumple con su objetivo, sin embargo, como no es posible verificar el resultado esperado y teniendo en cuenta los casos anteriores, es muy probable que el algoritmo no esté trabajando de la mejor forma.

```
[0 50 80 101 801 1601 1633 1677 1897 1957 1999 2559 2759 2819 3011 3459 3543 3549; 0 0 150 255 37
50 4770 4806 5026 5838 5619 5829 7899 7974 8259 8895 10419 10071 7086; 0 0 0 210 5100 4740 4716 5
058 5988 5514 5934 8394 7899 8289 8985 10689 9966 6986; 0 0 0 0 2100 4500 4596 4728 5388 5424 555
0 7230 7749 7929 8505 9849 9876 6926; 0 0 0 0 0 5600 5824 6132 7264 5388 5682 7908 7773 8100 8760
 10356 9849 6884; 0 0 0 0 0 0 3200 7600 12080 3288 7488 16968 7068 9348 11124 16068 7749 5484; 0
0 0 0 0 0 0 352 1520 888 1224 3528 3288 3636 4308 5940 5349 3884; 0 0 0 0 0 0 0 0 880 792 960 295
2 3132 3372 4020 5508 5253 3820; 0 0 0 0 0 0 0 0 0 660 1122 3660 3105 3516 4224 5964 5184 3732; 0
 0 0 0 0 0 0 0 0 840 4080 2580 3180 3996 6060 4605 3292; 0 0 0 0 0 0 0 0 0 0 1680 2280 2460 3
036 4380 4425 3172; 0 0 0 0 0 0 0 0 0 0 0 0 2800 3640 4880 7960 4380 3088; 0 0 0 0 0 0 0 0 0 0 0
0 0 2400 4160 8800 2700 1968; 0 0 0 0 0 0 0 0 0 0 0 0 0 0 960 3200 2100 1568; 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 5376 1920 1448; 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1344 1064; 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 168; 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[-1 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17; -1 -1 2 3 3 3 3 7 3 3 10 10 3 3 3 3 3 2; -1 -1 -1
3 3 3 3 3 3 3 10 10 3 3 3 3 3 3; -1 -1 -1 -1 4 5 6 7 8 7 10 11 10 13 14 15 10 4; -1 -1 -1 -1 -1 5
 6 7 7 5 10 10 10 10 10 10 5 5; -1 -1 -1 -1 -1 -1 6 7 7 6 10 10 10 10 10 10 6 6; -1 -1 -1 -1 -1 -
1 -1 7 7 7 10 10 10 10 10 10 7 7; -1 -1 -1 -1 -1 -1 -1 -1 8 8 10 10 10 13 10 10 10 8; -1 -1 -1 -1
 -1 -1 -1 -1 -1 9 10 10 10 10 10 10 10 9; -1 -1 -1 -1 -1 -1 -1 -1 -1 10 10 10 10 10 10 10 10;
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 11 12 13 14 15 13 11; -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 12 13
13 13 12 12; -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 13 13 13 13 13; -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 14 15 14 14; -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 15 15 15; -1 -1 -1 -1 -1 -
1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 16 16; -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 17; -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
(((((A1A2)A3)(A4A5))((A6A7)(A8A9)))((((A10A11)A12)(A13A14))((A15A16)(A17A18))))
Mínima cantidad de mult. escalares: 3549
```

Figura 5: Resultados del quinto caso

5. Conclusiones

De acuerdo con los resultados obtenidos es posible indicar que el algoritmo es eficiente generando las búsquedas de las multiplicaciones escalares en cada uno de los casos. Sin embargo, cuando se va aumentando la cantidad de matrices a asociar, la parentización no se muestra de la manera más óptima. Es decir, en algunos casos no tiene en cuenta la cantidad de multiplicaciones escalares correcta y refleja un resultado no tan óptimo como el esperado.