Programming Techniques and Data Structures

M. Douglas McIlroy Editor

Chain Multiplication of Matrices of Approximately or Exactly the Same Size

NICOLA SANTORO

ABSTRACT: We present a different approach to finding an optimal computation order; it exploits both the difference between the size of the matrices and the difference between the number of nonzero elements in the matrices. Therefore, this technique can be usefully applied where the matrices are almost or exactly the same size. We show that using the proposed technique, an optimal computation order can be determined in time $O(n)$ if the matrices have the same size, and in time $O(n^3)$ otherwise.

1. INTRODUCTION

Consider the problem of evaluating the product of n matrices

$$M = M_1 \times M_2 \times \ldots \times M_n$$

where M_i is a $w_i \times w_{i+1}$ matrix. Since matrix multiplication is associative, the order in which the above chain multiplication is evaluated does not affect the final result; however, it can greatly affect the total number of performed operations [1, 5, 9]. It is therefore an important and practical problem to determine the optimal multiplication order, that is, the order that minimizes the number of performed operations. Several algorithms for finding an optimal or near-optimal computation order already have been presented [1–5, 7–9].

An important observation has to be made about the behavior of these algorithms. The amount of "savings" resulting from choosing the (near) optimal order is related to the difference in the sizes of the matrices: the greater this difference, the greater in general will be the amount of savings in terms of the number of operations to be performed. As a consequence, if the matrices are almost the same size (i.e., the difference in size is very small), then the ratio between the total number of operations required by each order can be very small. In the extreme case in which all matrices are the same size, none of the proposed algorithms can be applied. This fact follows from the assumption, made in all these algorithms, that the total number of operations required to multiply a $p \times q$ matrix by a $q \times r$ matrix is (proportional to) $p \cdot q \cdot r$.

We consider the cases where the matrices are almost or exactly the same size and show that some significant savings in the total number of operations can still be obtained by exploiting both the difference (if any) in the size of the matrices and the difference in the number of nonzero elements. We base our results on the observation that it is possible to multiply a $p \times q$ matrix with k nonzero elements by a $q \times r$ matrix with m nonzero elements in time (proportional to) $\min \{p \cdot m, r \cdot k\}$. It is easy to see that (using this multiplication technique to evaluate the product of two matrices) the multiplication order in which a chain product of matrices is evaluated can affect the total number of operations, even in the extreme case where all matrices are the same size.

In the following sections, we assume that matrix multiplication is performed using this algorithm. We

show that the optimal order can be determined in time $O(n)$ if the matrices are the same size, and in time $O(n^3)$ otherwise.

2. PRELIMINARY CONSIDERATIONS

Consider the product of n matrices

$$M = M_1 \times M_2 \times \cdots \times M_n \tag{1}$$

where M_i is a $w_i \times w_{i+1}$ matrix with m_i nonzero elements, $1 \le i \le n$.

The traditional methods for finding an optimal or near-optimal ordering to evaluate Eq. (1) assume that the used multiplication algorithm requires $w_{i-1} \times w_i \times w_{i+1}$ operations to multiply a $w_{i-1} \times w_i$ matrix by a $w_i \times w_{i+1}$ matrix. We refer to such multiplication algorithms as *classical* ones. It has been shown that if we choose an appropriate order for evaluating Eq. (1), there can be significant savings in the number of performed operations [1, 5, 9]. The size of the savings is strictly related to the difference in the size of the matrices. Let MIN_c and MAX_c denote the minimum and the maximum number of operations needed to evaluate Eq. (1), respectively (i.e., the cost associated with the optimal and the least-efficient multiplication order, respectively). The size of the savings is given by

$$\text{SAVING}_c = \frac{\text{MAX}_c - \text{MIN}_c}{\text{MIN}_c} \tag{2}$$

Let

$$p = \max_{1 \le i \le n+1} \{W_i\} \tag{3}$$

$$q = \min_{1 \le i \le n+1} \{W_i\} \tag{4}$$

and

$$k = p - q \tag{5}$$

Then

$$\text{MAX}_c \le np^3 \tag{6}$$

$$\text{MIN}_c \ge nq^3 \tag{7}$$

that is

$$\frac{\text{MAX}_c}{\text{MIN}_c} \le \frac{p^3}{q^3} \tag{8}$$

Therefore, the size of the savings is bounded as follows:

$$\text{SAVING}_c \le 3\frac{k}{q} + 3\frac{k^2}{q^2} + \frac{k^3}{q^3} \tag{9}$$

From relation (9), it follows that if all matrices have the same size (i.e., $k = 0$), then

$$\text{SAVING}_c = 0$$

and if the matrices have almost the same size [i.e., $k = O(1)$], then

$$\text{SAVING}_c = O\left(\frac{1}{q}\right).$$

An alternative algorithm to multiply a $w_{i-1} \times w_i$ matrix with m_{i-1} nonzero elements by a $w_i \times w_{i+1}$ matrix with m_i nonzero elements in time $\min(w_{i-1} \cdot m_i, w_{i+1} \cdot m_{i-1})$ can be obtained by integrating the "forward" multiplication algorithm described in [6, 10] by the "backward" multiplication algorithm described in [10]. If we assume that matrix multiplication is performed by this integrated algorithm, then it is easy to see that the total number of operations (the cost) needed to evaluate Eq. (1) according to a multiplication order depends not only on the size of the matrices but also on the number of nonzero elements in the matrices.

	MAX	MIN	SAVING
C	29,128	28,160	0.034
A	13,000	6,400	1.031

FIGURE 1. Maximum (MAX) costs, minimum (MIN) costs, and amount of savings (SAVING), using the classical (C) and the proposed (A) algorithms in the chain product of Example 1.

Let MIN_A and MAX_A denote the minimum and the maximum number of operations needed to evaluate Eq. (1) when the matrix multiplication is performed using this integrated algorithm, and let

$$r = \max_{1 \le i \le n} \{m_i\} \tag{10}$$

$$s = \min_{1 \le i \le n} \{m_i\} \tag{11}$$

Then

$$\text{MAX}_A \le prn \tag{12}$$

$$\text{MIN}_A \ge qsn \tag{13}$$

that is

$$\frac{\text{MAX}_A}{\text{MIN}_A} \le \frac{pr}{qs} \tag{14}$$

Therefore, the size of the savings in this case is bounded as follows:

$$\text{SAVING}_A \le \frac{r}{s}\left(1 + \frac{k}{q}\right) - 1 \tag{15}$$

In the extreme case where the matrices are all the same size (i.e., $k = 0$), we have

$$\text{SAVING}_A \le \frac{r}{s} - 1$$

That is, by exploiting the difference in the number of nonzero elements in the matrices, there is still a possibility of performing fewer operations by choosing an appropriate multiplication order.

If the matrices have almost the same size [i.e., $k = O(1)$], then

$$\text{SAVING}_A = O(r/sq)$$

Since $r \ge s$, we can expect SAVING_A to be greater than SAVING_c. Let us illustrate the practical relevance of the above discussion by means of a simple example.

Example

Consider the product of four matrices of almost the same size

$$M = M_1 \times M_2 \times M_3 \times M_4$$

where $w_1 = 20$, $w_2 = 22$, $w_3 = 22$, $w_4 = 22$, $w_5 = 20$ and $m_1 = 100$, $m_2 = 200$, $m_3 = 100$, $m_4 = 100$. Using classical techniques, we observe that

$$\alpha = [(M_1 \times M_2)(M_3 \times M_4)]$$

is an optimal multiplication order requiring only $\text{MIN}_c = 28{,}160$ operations; we also observe that

$$\beta = \{M_1 \times [(M_2 \times M_3) \times M_4]\}$$

is the least-efficient multiplication order, requiring $\text{MAX}_c = 29{,}128$ operations to be performed. Since the matrices have almost the same size, the size of the saving is relatively small (as expected); in fact, we have $\text{SAVING}_c = 0.034$.

At the same time, by applying the proposed technique, we find that α is the least-efficient multiplication order requiring $\text{MAX}_A = 13{,}000$ operations and that β is an optimal order requiring $\text{MIN}_A = 6{,}400$ operations. In addition, we have $\text{SAVING}_A = 1.031$. It is interesting to note that the least-efficient order found using the proposed technique still requires fewer operations than the optimal order found using the classical methods. The numerical results for this example are summarized in Figure 1.

3. OPTIMAL MULTIPLICATION ORDER

A multiplication order α for evaluating the chain product $M_i * M_{i+1} * \ldots * M_j$ can be defined as a parenthesization of the integers $i, i+1, \ldots, j$. The possible multiplication orders for evaluating Eq. (1) when $n = 5$ are shown in Figure 2. We denote the set of integers parenthesized in α, by $\langle\alpha\rangle$ and the size of this set by $|\alpha|$. Any multiplication order α, $\langle\alpha\rangle = \{i, \ldots, k, \ldots, j\}$, where $|\alpha| \geq 2$, has the property that there exist two multiplication orders α' and α'', $\langle\alpha'\rangle = \{i, i+1, \ldots, k\}$, and $\langle\alpha''\rangle = \{k+1, \ldots, j\}$, such that $\alpha = (\alpha'\alpha'')$. For example, if $\alpha = ((12)((34)5))$, then $\alpha' = (12)$ and $\alpha'' = ((34)5)$. In the following, unless otherwise specified, we consider only multiplication orders that are parenthesizations of the integers $1, \ldots, n$.

To each multiplication order α, we associate a cost function, cost (α), defined as the *minimum* number of operations needed *in the worst case* to evaluate Eq. (1) according to α. If the actual number of nonzero elements of a matrix is not known *a priori,* we consider it to be equal to the product of its dimensions. Obviously, the only matrices of unknown size are the results of partial products obtained while evaluating Eq. (1).

Formally, given a chain of matrix products

$$M_{i,j} = M_i \times M_{i+1} \times \cdots \times M_j$$

we denote the minimum number of operations needed in the worst case to evaluate $M_{i,j}$ by $C_{i,j}$.
It is easy to observe that

$$C_{i,j} = \min_{i \leq k < j} \{C_{i,k} + C_{k+1,j} + \min(w_i m_{k+1,j}, w_{j+1} m_{i,k})\} \quad (16)$$

where

$$m_{i,j} = \begin{cases} m_i & \text{if } i = j \\ w_i w_{j+1} & \text{otherwise} \end{cases} \quad (17)$$

(((((12)3)4)5) (((1(23))4)5) (1(2(3(45))))
(((12)(34))5) ((1((23)4))5) ((1(2(34)))5)
((12)((34)5)) ((1(23))(45)) (1(2((34)5)))
(((12)3)(35)) (1((23)(45))) (1((2)34))5))
((12)(3(45))) (1(((23)4)5))

FIGURE 2. Multiplication orders for $n = 5$.

Using relations (16) and (17), an obvious $O(n^3)$ dynamic programming algorithm can be devised.

4. MATRICES OF SAME SIZE

Let us now consider the case when all matrices are the same size; that is, $w_i = w$ for all $i = 1, \ldots, n+1$. As discussed in Section 2, we can expect, even in this case, a reduction in the total number of operations by choosing an appropriate multiplication order.

To find the optimal multiplication order when all matrices are the same size, we can obviously use the $O(n^3)$ technique discussed in the previous section. However, we show that it is possible to develop a $O(n)$ algorithm to find the optimal order for this particular case.

The cost, cost (α), associated with the multiplication order α has been defined in Section 3 to be the minimum number of operations needed *in the worst case* to evaluate Eq. (1) according to α. This was done because in the evaluation of the actual cost of some multiplication orders we have to consider some variables that are unknown *a priori,* but for which there is an (obtainable) upper bound. It is easy to observe that the only orders where this problem occurs are those in which two or more subproducts of Eq. (1) are evaluated independently of each other.

Let us now introduce a particular class of multiplication orders. A *linear order* $\pi = (i_1, i_2, \ldots, i_n)$ is a permutation of the integers $1, 2, \ldots, n$ such that, for all $j = 1, \ldots, n$

$$i_j = \min_{k<j} \{i_k\} - 1 \quad \text{or} \quad i_j = \max_{k<j} \{i_k\} + 1 \quad (18)$$

To each linear order π_i we can associate a multiplication order α_i in a natural way

$$\alpha_i = ((\ldots((i_1 \cdot i_2) \cdot i_3) \ldots) \cdot i_n) \quad (19)$$

where, for each subsequence $\beta^j = ((\ldots((i_1 \cdot i_2) \cdot i_3) \ldots) \cdot i_j)$, $1 \leq j < n$,

$$(\beta^j \cdot i_{j+1}) = \begin{cases} (\beta^j i_{j+1}) & \text{if } i_j < i_{j+1} \\ (i_{j+1} \beta^j) & \text{otherwise} \end{cases} \quad (20)$$

We call α_i a *linear multiplication order,* and we refer to $(i_1 \cdot i_2)$ as the *innermost product* of α_i. Informally, a multiplication order α_i is linear if no subproducts can be evaluated independently of each other when evaluating Eq. (1) according to α_i. The possible linear multiplication orders for $n = 5$ are shown in Figure 3. Given a linear order $\pi_i = (i_1, i_2, \ldots, i_n)$, the cost of the associated linear multiplication order α_i can be easily deter-

((((12)3)4)5) (1((2(34))5))
(((1(23))4)5) ((1(2(34))5)
((1((23)4)5) (1(2((34)5)))
(1(((23)4)5)) (1(2(3(45))))

FIGURE 3. Linear multiplication orders for $n = 5$.

$A_1 = \{((((12)3)4)5)\}$

$A_2 = \{(((1(23))4)5), ((1((23)4))5), (1(((23)4)5))\}$

$A_3 = \{(1((2(34))5)), ((1(2(34))5), (1(2((34)5))))\}$

$A_4 = \{(1(2(3(45))))\}$

FIGURE 4. Partitions of the linear multiplication orders for $n = 5$.

mined. In fact, from relations (16) and (19), it follows that

$$\text{cost}(\alpha_i) = w\left[\sum_{j=1}^{n} m_j - \max(m_{i_1}, m_{i_2})\right] \qquad (21)$$

We will use this relation to prove the following

LEMMA.
Let all matrices have the same size. Then, for any multiplication order β, there exists a linear multiplication order α such that

$$\text{cost}(\beta) \geq \text{cost}(\alpha)$$

We will prove the lemma by induction on $|\beta|$. The lemma is obviously true for $|\beta| \leq 3$. Let it be true for $|\beta| = m - 1$; then, we will show that it holds for $|\beta| = m$. If β is linear, then the lemma trivially holds. Let β be nonlinear. By definition, there exist two multiplication orders, β' and β'' such that $\beta = (\beta'\beta'')$ with $m > |\beta'| \geq 1$ and $m > |\beta''| \geq 1$; by inductive hypothesis, there exist two linear multiplication orders, σ' and σ'' with $\langle\sigma'\rangle = \langle\beta'\rangle$ and $\langle\sigma''\rangle = \langle\beta''\rangle$ such that $\text{cost}(\sigma') \leq \text{cost}(\beta')$ and $\text{cost}(\sigma'') \leq \text{cost}(\beta'')$. Let $|\beta'| = |\sigma'| = j$, and let $(x \cdot x + 1)$ and $(y \cdot y + 1)$ be the innermost products of σ' and σ'', respectively. Since both σ' and σ'' are linear, then [relation (21)]

$$\text{cost}(\sigma') = w\left[\sum_{i=1}^{j} m_i - \max(m_x, m_{x+1})\right]$$

and

$$\text{cost}(\sigma'') = w\left[\sum_{i=j+1}^{n} m_i - \max(m_y, m_{y+1})\right]$$

Without loss of generality, let $|\sigma'| \geq |\sigma''|$. For the multiplication order $\sigma = (\sigma'\sigma'')$, we have

$$\text{cost}(\sigma) \leq \text{cost}(\beta)$$

If $|\sigma''| = 1$, then σ is linear and the lemma is proved. Otherwise, since the values of $m_{1,j}$ and $m_{j+1,n}$ are not known *a priori*, then

$$\text{cost}(\sigma) = \text{cost}(\sigma') + \text{cost}(\sigma'') + w^3$$

Consider now the linear multiplication order

$$\alpha = ((\ldots((\sigma'\ j + 1)\ j + 2)\ldots n)$$

We have

$$\text{cost}(\alpha) = w\left[\sum_{i=1}^{n} m_i - \max(m_x, m_{x+1})\right]$$

Since $\max(m_x, m_{x+1}) \leq w^2$, then

$$\text{cost}(\alpha) \leq \text{cost}(\sigma)$$

which proves the lemma.

As a consequence of the lemma, to determine the optimal order, we need only to consider linear multiplication orders. Another important property is that linear multiplication orders can be easily partitioned into sets, all the elements of the same set having the same cost. Let A_i denote the set of all the linear multiplication orders having $(i \cdot i + 1)$ as the innermost product, $1 \leq i < n$ (Figure 4). Then, from relation (21), it follows that

$$\forall \alpha_j, \alpha_k \varepsilon A_i \qquad \text{cost}(\alpha_j) = \text{cost}(\alpha_k) \qquad (22)$$

That is, we can find the optimal order by simply evaluating the cost of only one linear multiplication order for each set A_i, $1 \leq i < n$, and then finding the minimum.

Thus, the cost of the optimal order is given by

$$\text{optimal cost} = \min_{1 \leq i < n} \left\{ w\left(\sum_{j=1}^{n} m_j - \max(m_i, m_{i+1})\right)\right\}$$

$$= w\left(\sum_{j=1}^{n} m_j - \max_{1 \leq i \leq n} \{m_i\}\right)$$

In other words, the problem of finding the optimal order when the matrices have the same size can be reduced to the problem of finding the maximum number of nonzero elements in the matrices. In fact, let k be an index such that

$$m_k = \max_{i \leq i \leq n} \{m_i\}$$

then the linear multiplication order

$$\alpha = (1(\ldots(k - 1((\ldots((kk + 1)k + 2)\ldots n))\ldots))$$

is clearly optimal with cost

$$\text{cost}(\alpha) = w \sum_{i \neq k} m_i$$

Therefore, if the matrices are all the same size, the optimal multiplication order can be found in time $O(n)$.

5. CONCLUSION

We have presented a different approach to determining an optimal multiplication order for a chain of matrix products. This approach exploits both the difference between the size of the matrices and the difference between the number of nonzero elements in the matrices; it is therefore useful when the matrices to be multiplied are almost or exactly the same size. We have shown that, using this approach, an optimal order can be determined in time $O(n)$ if the matrices are the same size, and in time $O(n^3)$ otherwise.

We base our results on the observation that it is possible to multiply a $p \times q$ matrix with k nonzero ele-

ments by a $q \times r$ matrix with m nonzero elements in time (proportional to) $\min(p \cdot m, r \cdot k)$. The multiplication technique assumed in this paper uses a linked representation for matrices; therefore, our algorithm is particularly suited for sparse matrices. On the other hand, contiguously stored matrices must be reconfigured in a linked representation; however, the determination of an optimal order is done independently of the storage configuration: the actual restructuring (if needed) will be done afterwards, and only if the savings in the number of operations justifies it. It should be pointed out that the restructuring of a $p \times q$ matrix can be done in time (proportional to) $p \cdot q$.

Finally, the proposed technique should be viewed more as a complement than as an alternative to existing methods, to be used in all those cases where the other techniques cannot be employed efficiently.

REFERENCES

1. Aho, A.V., Hopcroft, J.E., and Ullman, J.D. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, MA, 1974.
2. Chandra, A.K. Computing matrix chain product in near optimal time. Rep. RC-5626, Thomas J. Watson Res. Ctr., Yorktown Heights, NY, 1975.
3. Chin, F.Y. An $O(n)$ algorithm for determining a near optimal computation order of matrix chain product. *Commun. ACM 21*, 7 (July 1978), 544–549.
4. Deimel, L.E., and Lampe T.A. An invariance theorem concerning optimal computation of matrix chain products. Rep. TR79-14, North Carolina State Univ., Raleigh, NC, 1979.
5. Godbole, S.S. An efficient computation of matrix chain products. *IEEE Trans. Computrs. C-22*, 9 (Sept. 1973), 864–866.
6. Gustavson, F.G. Two fast algorithms for sparse matrices: multiplication and permuted transposition. *ACM Trans. Math. Softw. 4*, 3 (Sept. 1978), 250–269.
7. Hu, T. C., and Shing, M. T. Computation of matrix chain product (Abstract). *Am. Math. Soc. 1*, 3 (April 1980), 336.
8. Hu, T. C., and Shing, M. T. Some theorems about matrix multiplication (Extended Abstract). *Proc. 21st Symp. Foundations of Computer Science.* Syracuse, NY, Oct. 1980, 28–35.
9. Muraoka, V., and Kuck, D.J. On the time required for a sequence of matrix products. *Commun. ACM, 16*, 1 (Jan. 1973), 22–26.
10. Santoro, N. Four $O(n^2)$ multiplication methods for sparse and dense Boolean matrices. *Proc. 10th Conf. Numerical Mathematics and Computing.* Winnepeg, Manitoba, Oct. 1980, 241–253.

CR Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*

General Term: Algorithms

Additional Key Words and Phrases: matrix chain product, sparse matrices, linear multiplication order.

Received 3/81; revised 5/81; accepted 6/83

Author's Present Address: Nicola Santoro, Distributed Computing Group, School of Computer Science, Carleton University, Ottawa, Ontario K1S 5B6 Canada