EVALUATION OF MATRIX CHAIN PRODUCTS ON PARALLEL SYSTEMS†

HEEJO LEE*, JONG KIM*, SUNGJE HONG*, and SUNGGU LEE**

*Dept. of Computer Science and Engineering
**Dept. of Electronic and Electrical Engineering
Pohang University of Science and Technology
San 31 Hyoja Dong, Pohang 790-784, KOREA
E-mail : {heejo,jkim}@postech.ac.kr

ABSTRACT

The problem of finding an optimal product sequence for sequential multiplication of matrices (the matrix chain ordering problem, MCOP) is well-known and has been studied for a long time. In this paper, we consider the problem of finding an optimal product schedule for evaluating a chain of matrix products on a parallel computer (the matrix chain scheduling problem, MCSP). The difference between MCSP and MCOP is that MCOP considers a product sequence for single processor systems and MCSP considers a sequence of concurrent matrix products for parallel systems. The approach of parallelizing each matrix product after finding an optimal product sequence for single processor systems does not always guarantee a minimal evaluation time since each parallelized matrix product may use processors inefficiently. We introduce a processor scheduling algorithm for MCSP which attempts to minimize the evaluation time of a chain of matrix products on a parallel computer, even at the expense of a slight increase in the total number of operations. Experiments on Fujitsu AP1000 multicomputer show that the proposed algorithm decreases the time required to evaluate a chain of matrix products in actual parallel systems.

Keywords – Processor scheduling, matrix chain product, dynamic programming, parallel matrix multiplication, matrix chain scheduling problem.

1 INTRODUCTION

Matrix multiplication is a computation intensive part of many commonly used scientific computing applications. In many applications such as robotics, machine control, and computer animation, a chain of matrices is multiplied consecutively. In the evaluation of a chain of matrix products with n matrices $\mathcal{M} = M_1 \times M_2 \times \cdots \times M_n$, where M_i is an $m_i \times m_{i+1} (m_i \geq 1)$ matrix, the product sequence greatly affects the total number of operations required to evaluate $\mathcal{M}$, even though the final result is the same for all product sequences by the associative law of matrix multiplication. An arbitrary product sequence of matrices may be as bad as $\Omega(T_{opt}^3)$ where T_{opt} is the minimum number of operations required to evaluate a chain of matrix products [1]. The matrix chain ordering problem (MCOP) is the problem of finding a product sequence for a set of matrices such that the total number of operations is minimized.

An exhaustive search to find an optimal solution for MCOP is not a good strategy since the number of possible product sequences of a chain of matrix products with n matrices is $\Theta(4^n/n^{3/2})$, which is known as the *Catalan number* [2]. Let us refer to the time required to find a product sequence for a chain of matrices as the *ordering time* and the time required to execute the product sequence as the *evaluation time*. There are many works reported for solving MCOP and for reducing the ordering time. MCOP was first reported by Godbole [3] and solved using dynamic programming in $O(n^3)$ time. Chin [4] suggested an approximation algorithm which runs in $O(n)$ time for finding a near-optimal sequence. The optimal sequential algorithm, which runs in $O(n \log(n))$ time, was given by Hu and Shing [5, 6]. This algorithm solves MCOP by solving the equivalent problem of finding an optimal triangulation of a convex polygon.

Many parallel algorithms to reduce the *ordering time* have been studied using the dynamic programming method [7, 8, 9, 10] and the convex polygon triangulation method [11, 12]. Bradford *et al.* [10] proposed a parallel algorithm based on dynamic programming which runs in $O(\log^3(n))$ time with $n/\log(n)$ processors on the CRCW PRAM model. Czumaj [11] proposed a $O(\log^3(n))$ time algorithm based on the triangulation of a convex polygon which runs with $n^2/\log^3(n)$ processors on the CREW PRAM. Ramanan [12] gives an algorithm which runs in $O(\log^4(n))$ time using n processors. Most of the recently proposed parallel algorithms run in polylog time with a linear number of processors.

Now let us consider the *evaluation time* of a chain of matrix products. In a single processor system, the evaluation of a chain of matrix products by the optimal product sequence guarantees the minimum evaluation time since the sequence guarantees the minimum number of operations. However, in parallel systems, parallel computation of each matrix product with the product sequence found for the minimum number of

† This research was supported in part by NON DIRECTED RESEARCH FUND, Korea Research Foundation.

operations does not guarantee the minimum evaluation time. This is because that the evaluation time in parallel systems is affected by various factors such as dependencies among tasks, communication delays, and processor efficiency.

In this paper, we formally present the problem of finding the matrix product schedule for parallel systems (MCSP) and analyze the problem complexity of MCSP. We propose an algorithm which finds a matrix product schedule that, while slightly increasing the number of operations, decreases the evaluation time of a chain of matrix products by finding sets of matrix products that can be executed concurrently.

This paper is organized as follows. Section 2 presents the formal description of the processor scheduling problem for a chain of matrix products and shows that the given problem is NP-Hard. In Section 3, we propose a matrix chain scheduling algorithm which dramatically reduces the evaluation time of a chain of matrix products by using processors efficiently in parallel systems and analyze the algorithm. In Section 4, we present the performance of the proposed method and compare with that of sequential matrix products ordered by the optimal product sequence for MCOP using experiments on the Fujitsu AP1000 parallel system. Finally, in Section 5, we summarize and conclude the paper.

2 MATRIX CHAIN SCHEDULING PROBLEM

2.1 NOTATIONS

- P : the number of processors in a parallel system.
- $\mathcal{M}$: a matrix chain product with n matrices, i.e., $\mathcal{M} = M_1 \times M_2 \times \cdots \times M_n$.
- M_i: an $m_i \times m_{i+1}$ matrix ($m_i \geq 1$, $1 \leq i \leq n$).
- L : a product sequence tree for a matrix chain $\mathcal{M}$.
- $L_{i,j}$: the sequence subtree of L for $(M_i \times \cdots \times M_j)$.
- C : the minimum amount of computation for evaluating $\mathcal{M}$.
- $c_{i,j,k}$: the amount of computation required for multiplying $m_i \times m_j$ and $m_j \times m_k$ matrices, $(m_i m_j m_k)$.
- ΔC : the amount of increased computation by modifying the current sequence tree.
- $p_{i,j}$: the number of processors assigned for evaluating $(M_i \times \cdots \times M_j)$.
- $T_{i,j}(p_{i,j})$: the execution time for evaluating $(M_i \times \cdots \times M_j)$ on $p_{i,j}$ processors.
- Φ: the execution time of single matrix product.

2.2 PROBLEM DESCRIPTION

We consider the problem of finding the optimal schedule with minimum evaluation time of $\mathcal{M} = M_1 \times M_2 \times \cdots \times M_n$ on a P processor parallel system. The number of operations for multiplying a matrix A of size $m_i \times m_j$ by a matrix B of size $m_j \times m_k$ is $m_i m_j m_k$. Many parallel algorithms for matrix multiplication have been developed in various parallel architectures [13]. The execution time of matrix multiplication depends on the algorithm used and the architecture on which the algorithm runs. However, for more discussion, we assume that the execution time of matrix multiplication is proportional to the number of operations on a processor. Therefore, for multiplying A by B matrices with p processors, the execution time $\Phi(m_i, m_j, m_k, p)$ can be approximated as follows.

$$\Phi(m_i, m_j, m_k, p) \approx \begin{cases} \frac{m_i m_j m_k}{p} & \text{if } 1 \leq p \leq m_i m_k \\ \frac{m_i m_j m_k}{p} \log(\frac{p}{m_i m_k}) & \text{otherwise} \end{cases}$$

When p_{ij} processors are allocated for evaluating $(M_i \times \cdots \times M_j)$, the evaluation time consists of two parts : the partial matrix chain evaluation time and the single matrix product execution time. Two partial matrix chains are $(M_i \cdots M_k)$ and $(M_{k+1} \cdots M_j)$ for some k $(i \leq k < j)$. The evaluation time of two matrix product chains is dependent on the evaluation method. One method is to evaluate sequentially $(M_i \cdots M_k)$ and $(M_{k+1} \cdots M_j)$ using all available processors in p_{ij}. The other method is to evaluate both $(M_i \cdots M_k)$ and $(M_{k+1} \cdots M_j)$ concurrently by partitioning p_{ij} into $p_{i,k}$ and $p_{k+1,j}$ such that $p_{i,k} + p_{k+1,j} \leq p_{ij}$. The minimum evaluation time $T_{i,j}(p_{i,j})$ of $(M_i \cdots M_j)$ on $p_{i,j}$ processors is found from the following recurrence relation.

$$\begin{aligned} T_{i,j}(p_{i,j}) \approx \min_{i \leq k < j} \Big(& T_{i,k}(p_{i,j}) + T_{k+1,j}(p_{i,j}) + \\ & \Phi(m_i, m_{k+1}, m_{j+1}, p_{i,j}), \\ & \max\Big(T_{i,k}(p_{i,k}), T_{k+1,j}(p_{k+1,j})\Big) + \\ & \Phi(m_i, m_{k+1}, m_{j+1}, p_{i,j}) \Big) \end{aligned}$$

The problem of finding the schedule which results in the minimum evaluation time $T_{1,n}(P)$ is a problem of finding the best schedule $k_{i,j}$ for $(M_i \cdots M_j)$ with the processor allocation p_{ij} to $L_{i,j}$. Therefore, the MCSP is defined as follows.

MCSP: *find the product sequence for evaluating a chain of matrix products and the processor schedule for the sequence, where the evaluation time is minimized on a parallel system.*

2.3 MCSP COMPLEXITY

Consider the case in which there are sufficient processors for multiplying any number of matrices concurrently. We assume that, for each matrix product which multiplies an $m_i \times m_j$ matrix by an $m_j \times m_k$ matrix, $m_i m_j m_k$ processors are allocated and the computation time is $\log(m_j)$. The problem can be solved in polynomial time using dynamic programming of the recurrence relation.

$$T_{i,j} = \min_{i \leq k < j} \Big(\max(T_{i,k}, T_{k+1,j}) + \log(m_{k+1}) \Big)$$

Therefore, in the case of infinitely many available processors, the problem of finding the schedule for evaluating $\mathcal{M}$ with the minimum execution time can be solved using dynamic programming in polynomial time, i.e., $O(n^3)$. However, in general, the number of available

processors is fixed and not sufficient to allocate the requested number of processors for each product.

Now, let us consider the case when there are P processors in a system. The complexity of MCSP is shown in the following Theorem.

Theorem 1: *MCSP* is a NP-hard problem.

In this paper, the proofs for all theorems and lemmas have been omitted due to lack of space. The interested reader can obtain the proofs from [14]

Since the problem of finding an optimal scheduling is NP-Hard, we propose an approximation algorithm in Section 3. The algorithm enhances the performance of evaluating an n-matrix product chain on a parallel system by partitioning the parallel system and concurrently executing several matrix products; this also enhances the overall efficiency of the system.

3 MATRIX CHAIN SCHEDULING ALGORITHM

The proposed scheduling algorithm consists of three stages. First, the algorithm finds the optimal product sequence for MCOP. Next is the top-down processor assignment stage. In this stage, processors are partitioned and assigned to each subtree proportionally according to its computation amount to balance the evaluation time of both left and right partial matrix product chains. The third stage is the bottom-up execution stage that executes products independently from the leaf and tries to modify the product sequence to enhance the concurrency so as to reduce the evaluation time of $\mathcal{M}$. This is done by finding the points that change the product sequence but do not increase the number of operations excessively.

3.1 OPTIMAL PRODUCT SEQUENCE BY MCOP

The product sequence of $\mathcal{M}$ determines the number of operations to be executed in single processor systems. In parallel systems, the number of operations is not the sole deciding factor of the evaluation time, but affects it greatly nonetheless. Hence, our scheduling algorithm begins with the optimal product sequence found for MCOP. There were many works reported for finding the optimal product sequence which guaranteed the minimum number of operations for any chain of matrix products. The optimal product sequence can be found in $O(n\log(n))$ time using a sequential algorithm [5, 6]. Many parallel algorithms have been studied [10, 11] which run in polylog time. Therefore, using these parallel algorithms, it is possible to find the optimal product sequence within polylog time on P processor systems.

Let us assume that the sequence and the computation amount found by MCOP is stored in two tables, $S[n][n]$ and $W[n][n]$ respectively. $W[i][j]$ has the minimum number of operations for evaluating $L_{i,j}$, and $S[i][j]$ has the matrix index for partitioning the matrix chain $(M_i \times \cdots \times M_j)$.

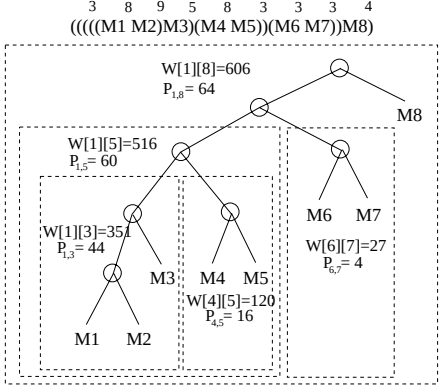

Figure 1: Top-down processor assignment.

3.2 TOP-DOWN PROCESSOR ASSIGNMENT

In the top-down processor assignment stage, the number of processors proportional to the computation amount of a subtree is assigned to minimize the completion time of two partial matrix chains. If $p_{i,j}$ processors were assigned to $L_{i,j}$, then $p_{i,j} \times \frac{W[i][S[i][j]]}{(W[i][S[i][j]]+W[S[i][j]+1][j])}$ processors are assigned to the subtree $L_{i,S[i][j]}$ and $p_{i,j} \times \frac{W[S[i][j]+1][j]}{(W[i][S[i][j]]+W[S[i][j]+1][j])}$ processors are assigned to the subtree $L_{S[i][j]+1,j}$.

For example, given a chain of 8 matrices with dimensions $\{3, 8, 9, 5, 8, 3, 3, 3, 4\}$ on a 64 processor system, processors are assigned as in Fig. 1.

3.3 BOTTOM-UP CONCURRENT EXECUTION

After assigning processors to each subtree, the matrix products are executed concurrently and independently from the leaf products. Thus, the performance is improved by the concurrent execution which results in the improvement of processor efficiency. However, there are cases in which some processors stay idle. When there are idle processors in the execution of $L_{i,j}$, we try to modify the product sequence to use these idle processors.

When there are idle processors in the execution of $L_{i,j}$, ancestors of the leaf node of $L_{i,j}$ are traced in order to find a candidate for concurrent execution. This upward trace continues until a suitable candidate or a sibling which is not a leaf node is found. For example, let us consider the executing sequence tree $L_{1,9}$ shown in Fig. 2. The figure represents $(((M_1(M_2(M_3(((M_4M_5)M_6)M_7))))M_8)M_9)$. In the execution of (M_4M_5), the possible candidates for concurrent execution are $(M_1M_2), (M_2M_3), (M_6M_7), (M_8M_9)$. There are other types of candidate products which are not considered in this paper like (M_7M_8). Since such cases result in more modifications to the optimal sequence with no obvious benefit over other candidates, we do not consider these kinds of products.

When we modify the product sequence to execute

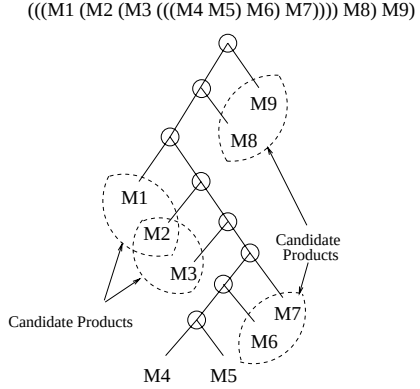

Figure 2: Candidate products on a sequence tree.

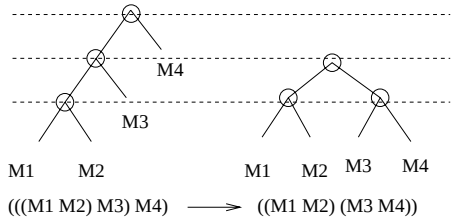

Figure 3: A sequence modification.

candidate products concurrently in the current execution phase, then there is some loss due to an increase in the number of operations. Therefore, we have to check whether the modification is beneficial or not.

Consider a matrix chain with 4 matrices which needs three matrix product to get the final result. Assume that the optimal product sequence of this matrix chain for MCOP is $(((M_1M_2)M_3)M_4)$, as shown in Fig. 3. When the sequence is modified to $((M_1M_2)(M_3M_4))$, the number of computations required changes from $C = c_{1,2,3} + c_{1,3,4} + c_{1,4,5}$ to $C' = c_{1,2,3} + c_{3,4,5} + c_{1,3,5}$. Therefore, the amount of increased computation ΔC is as follows.

$$\Delta C = c_{3,4,5} + c_{1,3,5} - c_{1,3,4} - c_{1,4,5}$$

In general, when we have a product sequence like Fig. 4, the amount of increased computation for multiplying $M_y \times M_{y+1}$ concurrently with $M_x \times M_{x+1}$ is as follows.

$$\Delta C = m_{y+1}m_{y+2}(m_y - m_z) + m_z m_y (m_{y+2} - m_{y+1})$$

In this equation, m_z represents the row of the intermediate matrix (or matrix M_z itself) that is going to be multiplied with the result of $M_y \times M_{y+1}$. In other words, there is a left parenthesis to the left of matrix M_z that matches the right parenthesis on the right

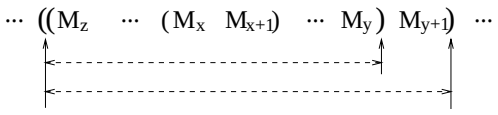

Figure 4: A snapshot of product sequence.

side of matrix M_{y+1}. In case $y < z$, the amount of increased computation is as follows.

$$\Delta C = m_{y+1}m_{y+2}(m_y - m_{z+1}) + m_{z+1}m_y(m_{y+2} - m_{y+1})$$

Finding m_z (or m_{z+1}), which is very important for the analysis of ΔC, can be done by traversing the sequence tree L. If both M_y and M_{y+1} are right children, then M_z is searched by traversing the left child recursively from the parent node of M_y. Similarly, if both are left children, then M_z is searched by traversing the right child from the parent node of M_y.

Lemma 1: The evaluation time is reduced by modifying the sequence tree L when the candidate product (M_yM_{y+1}) satisfies $\Delta C < \min(\Phi(m_x, m_{x+1}, m_{x+2}, m_xm_{x+2}) \times (p_{i,j} - m_xm_{x+2}), c_{y,y+1,y+2})$.

If a candidate product (M_yM_{y+1}) satisfies Lemma 1, then it would be better to change the product sequence $L_{i,j}$ to multiply the candidate product concurrently with (M_xM_{x+1}). This means that the unallocated idle processors can do more work than the increased computation required by the change in the product sequence.

When the candidate product is found, the subtree $L_{i,j}$ is modified and the processors $p_{i,j}$ are redistributed among the products in $L_{i,j}$ (including the candidate product). Also, processors are allocated proportionally to each product. Thus, the overall system performance is enhanced by increasing the processor efficiency.

3.4 THE PROPOSED SCHEDULING ALGORITHM

The proposed scheduling algorithm for evaluating a matrix chain product is formulated as follows.

Two-Pass Matrix Chain Scheduling Algorithm

Stage–1 *MCOP*

1. Find the optimal product sequence by a parallel algorithm for MCOP.
2. Generate the sequence tree L.

Stage–2 *Top-Down Processor Assignment*

1. Initialize $i = 1$, $j = n$, $p_{i,j} = P$.
2. If i is not $S[i][j]$, then allocate $p_{i,j} \times W[i][S[i][j]]/(W[i][S[i][j]] + W[S[i][j]+1][j])$ processors to $L_{i,S[i][j]}$.
3. If j is not $S[i][j]+1$, then allocate $p_{i,j} \times W[S[i][j]+1][j]/(W[i][S[i][j]] + W[S[i][j]+1][j$