A Novel Approach for solving Matrix Chain Product

Pradhan Rajeshkumar I[1] Sanjay M. Shah[2]
[1]P. G. Scholar [2]Professor
[1,2]Department of Computer Engineering
[1,2]Government Engineering College, Modasa, India

Abstract— The Matrix chain product or (Matrix Chain multiplication) problem is the most famous illustration of dynamic approach of algorithm design. The matrix chain product problem tries to obtain the optimal sequence for doing a series of operations. The need for solving Matrix Chain Product problem is given a chain of matrices to multiply, obtain the fewest number of multiplications required to calculate the product. The original time complexity of Matrix Chain Product which uses Dynamic Programming to get the Multiplication cost is O(n3) which consumes more resources. Using Proposed novel approach we can reduce it to O(n2) which will save resource consumption and give us the result in lesser time.

Key words: Matrix Chain Product

I. INTRODUCTION

In recent past, lot of algorithms have been developed for matrix chain product[4][10].Many of these algorithms use dynamic programming which iteratively calculates the time needed to compute the minimum cost needed to multiply a specific sequence and save it. This dynamic approach is a powerful tool for algorithms and is used in many other problems like all pair shortest path (Floyd-Warshall algorithm), binary knapsack problem etc. In this paper an alternate approach has been described to calculate the minimum cost required to compute the chain product using a novel approach. The proposed method is named as Novel MCM (Matrix Chain Multiplication).

The paper is organized in following sections. In section II a brief introduction is given about what is matrix chain product problem and how to calculate the cost of multiplying the chain matrices. In section III, the traditional approach using dynamic solution is discussed. The next sections define the proposed algorithm Novel MCM and explain the working using some examples. The paper ends with some tables and graphs displaying computational results of the proposed method and comparing those with dynamic approach.

II. OUTLINE OF MATRIX CHAIN PRODUCT

Consider the problem of evaluating the product of *n* matrices

$$M=M_1*M_2*\dots\dots*M_n$$

Where M_i is a matrix of the order mxn and $M_{(i+1)}$ is a matrix of order nxp. The product N = $M_i*M_{(i+1)}$ is a mxp matrix. This N can be computed in time O(mnp). For example, let there be 4 matrices named A, B, C, D of the order (2x3), (3x4), (4x5), (5x6) respectively. Now M = A*B*C*D

Since matrix multiplication is associative, the order in which the above chain multiplication is evaluated does not affect the final result. The matrix can be multiplied in the following orders: ((AB)C)D, (AB)(CD), A((BC)D), (A(BC))D, A(B(CD)).

The problem is not actually to perform the multiplication, but to decide the order in which multiplications needs to be performed. Because this order in which the product of matrices is parenthesized affects the number of simple arithmetic operations needed to compute the product, or the efficiency [8]. Let us evaluate the number of arithmetic operations performed for all the above mentioned parenthesization

((AB)C)D = 2x3x4 + 2x4x5 + 2x5x6 = 124
(AB)(CD) = 2x3x4 + 4x5x6 + 2x4x6 = 192
A((BC)D) = 3x4x5 + 3x5x6 + 2x3x6 = 186
(A(BC))D = 3x4x5 + 2x3x5 + 2x5x6 = 150
A(B(CD)) = 4x5x6 + 3x4x6 + 2x3x6 = 228

Clearly the first method is more efficient in all. This gives a picture as the number of operations termed as scalar multiplications is affected by the order in which the product is computed. Thus, the number of scalar operations required depends on optimal parenthesis order. Now in order to determine the optimal parenthesis order, we can proceed in many ways. One is brute force method where we calculate the number of operations of all the possible parenthesis order and find the least amongst them. But it will be impractical and time consuming as the time complexity of such approach will be $O(2^n)$. The traditional approach i.e. dynamic solution [1][2][3], reduces this time of $O(2^n)$ further to $O(n^3)$which looks decent and of practical use.

III. DYNAMIC APPROACH

As described above, the cost of multiplying a chain of n matrices M1M2…..Mn depends on the order in which the n−1 multiplications are carried out. Here we will discuss in brief about the dynamic approach for the matrix chain multiplication problem as described in [1][7].

In general, the number of orderings is equal to the number of ways to place parentheses to multiply the n matrices in every possible way. Let f(n) be the number of ways to fully parenthesize a product of n matrices. Suppose we want to perform the multiplication

$$(M1M2 \ldots Mk) \times (Mk{+}1Mk{+}2 \ldots Mn).$$

Then, there are P(k) ways to parenthesize the first k matrices. For each one of the P(k) ways, there are P(n-k) ways to parenthesize the remaining n-k matrices, thus for a total of P(k)P(n - k) ways. Since k can be assumed any value between 1 and n – 1, the overall number of ways to parenthesize the n matrices is given by the summation:

$$P(n) = \begin{cases} 1, n = 1 \\ \sum_{k-1}^{n-1} P(k)P(n-k), n \geq \end{cases} = \Omega\left(4^n/_{n^{3/2}}\right) \quad (1)$$

(Using Catalan numbers and Stirlings' formula)[1].

In dynamic approach, the cost of multiplying two matrices together is done in a recursive way which can be represented as –

$$m[i][j] = \begin{cases} 0, \quad if\ i = j \\ \min\limits_{i \leq k < j} \begin{cases} m[i][k] + \\ m[k+1][j] + \\ P_{i-1}P_kP_j \end{cases} \quad (if\ i < j) \end{cases} \quad (2)$$

The algorithm for the same using dynamic approach can be found in [1]. Thus, time and space complexities of the algorithm are easy to calculate. For constant c > 0, the running time T(n) of algorithm is proportional to:

$$\mathbf{T(n)} = \sum_{x=2}^{n}\sum_{i=1}^{n}\sum_{k=1}^{n-x} \frac{\mathbf{cn^3 - cn}}{\mathbf{10}} = (\boldsymbol{n^3}) \quad (3)$$

Similarly space complexity of the algorithm is:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{cn^2 - cn}{n} = (\boldsymbol{n^2}) \quad (4)$$

IV. PROPOSED ALGORITHM USING NOVEL APPROACH

This section presents a solution for the problem to determine the minimum number of scalar multiplications performed for the matrix chain product problem using novel approach.

A. Proposed Flowchart

The proposed flowchart is shown in below figure

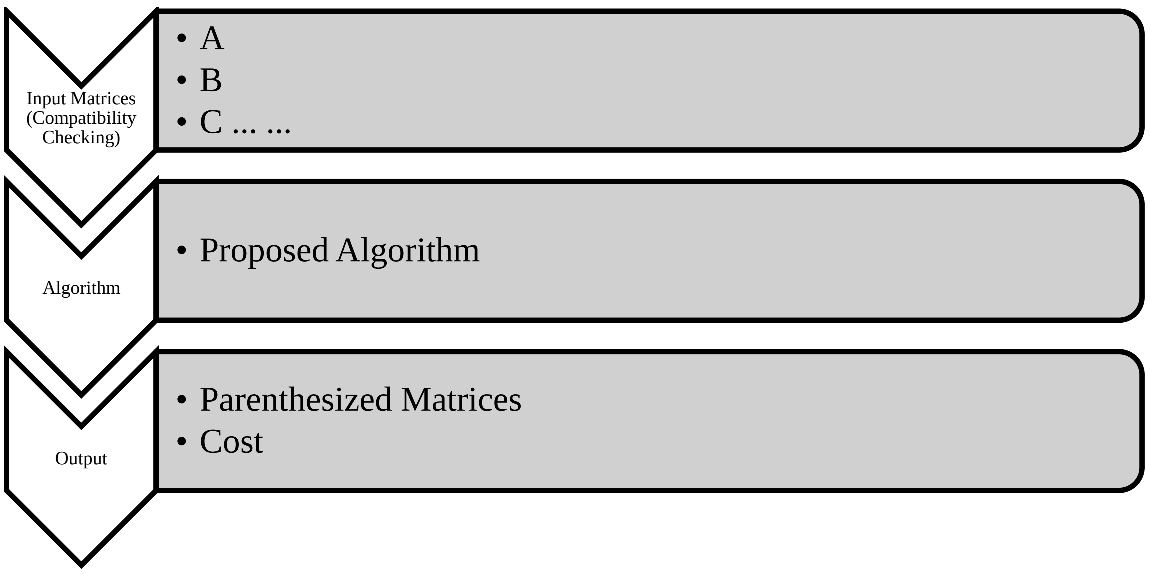

Fig. 1: Proposed Flowchart

As shown in above figure set of compatible matrices are given input to the proposed novel algorithm and the output of the algorithm is multiplication cost and parenthesized matrices.

B. Proposed Work Mathematically

Generally there are *(2nCn) / (n+1)* ways to parenthesize the matrices if we follow the brute force method. But here is my proposed mathematical approach to reduce the time complexity [4], [7].

- Let Matrices A, B, C, D, E of following given dimensions. $_{Am*n}$,B_{n*p} , C_{p*q}, D_{q*r} E_{r*s}
- Let's *C* is a variable which stores cost of the chained matrix multiplication. Initially *C=0*
- Find *min{m*p, n*q, p*r , q*s}* //Greedy Strategy
- Now suppose *{nq}* is the result of above *min* function.
- *C=C+ n*p*q*
- Now find *min{m*q, n*r , q*s}*
- Now suppose *{nr}* is the result of above *min* function.
- C=C+ n*q*r
- Now find *min{m*r , n*s}*
- Now suppose *{mr}* is the result of above *min* function.
- *C=C+ m*n*r + m*r*s;* Here the *C* gives us the required cost and (*(A((BC)D))E)* is the required parenthesization. Here O(n) time is required to find cost and parenthesization which is cost to find the *min* function.

1) Example

- Consider the chain of five matrices <A1, A2, A3, A4, A5> of dimension 3x4, 4x3, 3x6, 6x3, 3x7 respectively.
  Cost By:-
- Samsung & Amazon Method:-216
- Dynamic Programming:-180
- Now let's input the above matrices to the proposed algorithm.
- So according to proposed algorithm :-
- m=3,n=4,p=3,q=6,r=3,s=7
- Now *min{m*p, n*q, p*r , r*s}*
  *min{9,24,9,42} =9=m*p*
  *Cost=C=0+m*n*p=36*
  *Min{m*q, p*r, q*s}={18,9,42}=9=p*r*
  *Cost=C+p*q*r=36+54=90*
  *Min{m*r , p*s}={9,21}=9=m*r*
  *Cost=C+m*p*r+m*r*s=90+27+63=180*
- So by proposed algorithm also cost is 180 which is same as dynamic programming and parenthesization is (((A1A2)(A3A4)) A5)) .

C. Proposed Algorithm

Novel_MCM( ) :- determines the best order to multiply given matrices so you minimize the number of single element or scalar multiplications(Multiplication Cost) .It gives maximum time optimal solution or nearer to the optimal solution.

- N:- Given N number of matrices
- C:-Stores the total multiplication cost
- array p[1 .. N]:- Array p stores the dimension of metrics.

Suppose we want to find Matrix Chain Multiplication of

*A: 2*3*
*B: 3*4*
*C:4*2*
Then p stores 2,3,4,2

- *i,k:- Used for indexing purpose*

```
Novel_MCM(array p[1 .. N], int N)
{
        Initially C=0
        Loop:k = Find_min{p[1]*p[3],  p[2]*p[4], ... ,  p[N-2]*p[N]}
        // if p[k]*p[k+2] is minimum then   it returns "k"
        C=C + p[k]*p[k+1]*p[k+2]
        For i=k+1 to N // copies p[k+2] to p[k+1], and p[k+3]  to p[k+2], and so on..
        p[i]=p[i+1]
        N=N-1
        repeat till N>=3.
}
```

1) Find_min( ) :-

- min :- It stores the minimum value among all diagonally matrices multiplications

– Find_min( ) :- if p[j]*p[j+2] is minimum then it returns "j"
– j :- is index which loops through p[ ].
– Flag :-flag contains j where p[j]*p[j+2] is minimum

```
Find_min(int p[], int N)
{
        min= p[1]*p[3]
        for i=2 to N-2
        {
                If (min > p[i] * p[i+2])
                {
                min= p[i] * p[i+2]
                flag= i
                }
        }
        return flag
}
```

V. Result Analysis

A. Result Analysis

1) Complexity Analysis [7]

	Time Complexity	Space Complexity
Traditional Approach	$O(2^n)$	$O(n^2)$
Dynamic Programming	$O(n^3)$	$O(n^2)$
Greedy and D&C	$O(n^2)$ wc	O(n)
Novel_MCM	$O(n^2)$	O(n)

Table 1: Complexity Analysis

2) Time Complexity

The time complexity of Matrix Chain multiplication using novel approach is $O(n^2)$ because Find_min takes O(n) time .And we run Find_min for n times which is no. of matrix in our case.

Find_Min $= \sum_{j=0}^{n} O(1)$= O(n)

$$\text{Time Complexity} = \sum_{i=0}^{n} \text{Find\_Min} = \sum_{i=0}^{n} \sum_{j=0}^{n} O(1) = O(n^2)$$

So, Overall time Complexity of Matrix Chain multiplication using novel approach is $O(n^2)$.

3) Space Complexity

Here we use only one array of size n which used to store dimension of input matrices i.e. p[ ] .So, space complexity is O(n) of proposed algorithm.

4) Result Analysis for different dimensional matrices based on implementation in Turbo C

No. of Matrix	Sequence of Dimensions	Novel_ MCM M Cost*	Greedy and D&C M Cost* (Base Paper)	Dynamic Approach M Cost*
3	16,13,4,9	1408	1408	1408
4	23,21,2,24,21	2940	2940	2940
5	6,12,14,2,12,14	984	984	984
6	22,16,23,2,3,6,9	1980	1980	1980
7	12,10,12,10,7,5,18,15	4400	4400	4400
8	1,8,21,7,20,18,1,8,1	848	849	831
9	2,21,9,25,16,2,14,6,23,24	3336	3296	3236

Table 2: Multiplication Cost Comparison Table of different approaches

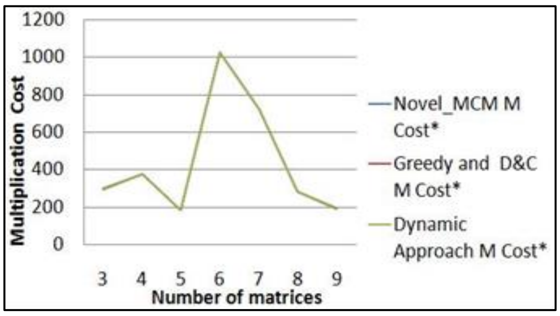

Fig. 2: Comparision of Multiplication Cost vs Number of matrices

5) Future work

- Matrix parenthesization.
- Finding the special cases for which proposed algorithm fails or work better by taking set of matrices in an excel sheet and giving them input to proposed algorithm, base paper algorithm ,dynamic approach. For this we will use any sophisticated framework i.e. visual studio.
- Consideration of run time cost in result analysis.

VI. CONCLUSION

The most popular method for solving Matrix Chain Product problem available in the market is Dynamic Programming right now [1], [3], [7].But it takes O(n3) time to get the chain matrix multiplication cost and O(n2) space complexity [7].Using the proposed algorithm we can bring down the time complexity of the problem to O(n2) and space complexity to O(n).

REFERENCES

[1] S. S. Godbole, "On Efficient Computation of Matrix Chain Products," IEEE Transactions on Computers, Vol. 100, No. 9, Pp. 864–866, 1973.

[2] S. A. Strate And R. L. Wainwright, "Parallelization Of The Dynamic Programming Algorithm For The Matrix Chain Product On A Hypercube," In IEEE Symposium On Applied Computing, 1990, Pp. 78–84.

[3] H. Nimbark, S. Gohel, And N. Doshi, "A Novel Approach For Matrix Chain Multiplication Using Greedy Technique For Packet Processing," Computer Networks And Information Technologies, Pp. 318–321, 2011.

[4] K. Nishida, Y. Ito, and K. Nakano, "Accelerating the Dynamic Programming for the Matrix Chain Product on the GPU," In IEEE 2nd International Conference On Networking And Computing, 2011, Pp. 320–326.

[5] X. Wang, D. Zhu, and J. Tian, "Efficient Computation Of Matrix Chain," In Proceedings Of The 8th International Conference On Computer Science And Education, 2013, Pp. 703–707.

[6] B. Ben Mabrouk, H. Hasni, And Z. Mahjoub, "Performance Evaluation Of A Parallel Dynamic Programming Algorithm For Solving The Matrix Chain Product Problem," In Ieee/Acs International Conference On Computer Systems And Applications, 2014, Pp. 109–116.

[7] R. Lakhotia, S. Kumar, R. Sood, H. Singh, And J. Nabi, "Matrix-Chain Multiplication Using Greedy And Divide-Conquer Approach," International Journal Of Computer Trends And Technology, Vol. 23, No. 2, Pp. 65–72, 2015.

[8] T. Iakymchuk, A. Rosado-Munoz, M. B. Mompean, J. V. F. Villora, And E. O. Osimiry, "Versatile Direct And Transpose Matrix Multiplication With Chained Operations: An Optimized Architecture Using Circulant Matrices," Ieee Transactions On Computers, Vol. 65, No. 11, Pp. 3470–3479, 2016.

[9] Chin, F.Y. "An O(N) Algorithm For Determining A Near Optimal Computation Order Of Matrix Chain Product." ACM , Pp . 544-549, 1978

[10] Nicola Santoro, "Chain Multiplication of Matrices of Approximately or Exactly the Same Size." Vol. 27, No. 2, Pp. 152-156, 1984.