ISBN: 978-1-948012-17-1

Ethics and Information Technology (ETIT)

DOI: <http://doi.org/10.26480/et.02.2020.27.30>

CrossMark

COMPARISON STUDY AND OPERATION COLLAPSING ISSUES FOR SERIAL IMPLEMENTATION OF SQUARE MATRIX MULTIPLICATION APPROACH SUITABLE IN HIGH PERFORMANCE COMPUTING ENVIRONMENT

Siddhartha Chatterjee^a, Soumitra De^b, Ahona Ghosh^c, Saikat Samanta^d^aDepartment of Computer Science and Engineering, Gargi Memorial Institute of Technology, Kolkata 700144, West Bengal, India.^bDepartment of Computer Science and Engineering, College of Engineering and Management, Kolaghat 721171, West Bengal, India.^cDepartment of Computational Science, Brainware University, Kolkata 700125, West Bengal, India.^dDepartment of Computer Science and Engineering, Bengal College of Engineering, Durgapur 713212, West Bengal, India*Corresponding Author Email: siddhartha.chatterjee31@gmail.com

This is an open access article distributed under the Creative Commons Attribution License CC BY 4.0, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

ARTICLE DETAILS

Article History:

Received 25 October 2020

Accepted 26 November 2020

Available online 03 December 2020

ABSTRACT

Multiplication between two square matrices is one of the fundamental computational approach in the domain of mathematics and computer science where it is fully recognized as a foremost technique for several interdisciplinary domain and sub domains like linear algebra, graph theory, multidimensional graphics, cryptographic computation, convolution neural network, deep learning, digital signal processing, medical image processing, steganography, relativity theory, quantum computing and many others. In a high-performance computing environment computational complexity analysis of matrix multiplication algorithm ensures a powerful paradox that takes a massive data processing approach to the world where problem solution feasibility comes down in respect of operation and time. In our paper we have analyzed different methods to multiply two square matrices (like 2×2 matrices) and their arithmetic complexity analysis in an asymptotic flavor along with operation collapsing issues through serial processing technique. We have analytically and experimentally explained and shown using MATLAB 9.3 simulator that fast matrix multiplication approach like Strassen and Winograd perform much better than conventional matrix multiplication algorithm especially for large amount of data.

KEYWORDS

Square matrix multiplication, Strassen matrix multiplication, Copper Smith-Winograd matrix multiplication, Serial processing.

1. INTRODUCTION

Square matrix multiplication concepts always stand up with fundamental operation from our childhood days to solve a problem in mathematics and now it is more essential in various fields of computational computing with respect of both practical and theoretical aspects. The proof can be determined in different fields of theoretical computer science research, applied mathematics, statistics, physics, bioinformatics, electronics and many other interdisciplinary areas where we have to frequently deal with large amount of data/information. Any type of computational applications in real world always need faster algorithm that can be solved within an optimum time. That's why computational speed for solving those applications mainly rely on the execution time of algorithm aside from other instantaneous requirement. So, when we are going to do something in a high-performance computing domain then maximum time apart from memory space, time requirement is one and only valuable resource for any algorithm by which we can get faster result along with effective performance. In this scenario, design, analysis and deep exploration for getting faster square matrix multiplication algorithm is profoundly required and that has already become the main purpose in computational computing research. This paper discusses about different types of matrix multiplication approach, its operation collapsing issues and asymptotic

time complexity analysis result using a serial processing system.

The paper remains organized according to different part discussed the following. Section 2 summarizes the methodology used in current literatures. A discussion based on the experimental analysis of the said methods is presented in section 3. And finally, Section 4 closes the analytical and comparison study on square matrix multiplication using sequential processor.

1.1 Related works

There has been a significant number of works in this domain. In previous study, stated that Strassen's matrix multiplication requires additional memory locations than the traditional algorithms as in Strassen's, additional arrays require to be formed (Mathew and Kumar, 2012). Among conventional algorithm, block matrix multiplication algorithm and Strassen's algorithm, the last one has shown better performance for large size matrices, whereas for small matrix, the recursion overhead resulted in inefficiency for Strassen's algorithm. In previous research, that proposed a Strassen-like method where additive complexity has been halved compared to Strassen's method (Bodrato, 2010). The proposed one can perform better for squaring, chain multiplications and general polynomial calculation. In this paper we are mainly trying to analyze and

Quick Response Code	This paper was presented at	Access this article online	
	International Conference on Contemporary Issues in Computing (ICIC-2020) - Virtual IETE Sector V, Salt Lake, Kolkata From 25th-26th July 2020	Website: www.intelcomp-design.com	DOI: 10.26480/et.02.2020.27.30

establish a comparative study in between naïve matrix multiplication approach and other two fast matrix multiplication approach like Strassen's and Winograd. In these consequences different multiplication algorithm plays an important role over the past decades. Until the late 1960's it was believed that to perform the multiplication of two $(n \times n)$ matrices, a cubic number of processes are mandatory, as the quicker algorithm that actually runs in $\Theta(n^3)$ time. In 1969, a study thrilled the researchers by delivering the first sub cubic time algorithm for product of matrix, consuming $\Theta(n^{2.807})$ time (Huss et al., 1996). After that in 1986, Strassen decreased the computational bound to 2.479. Three years later in 1989, Coppersmith and Winograd combined Strassen's technique and established the running time for square matrix multiplication in $\Theta(n^{2.376})$ time which has remained unchanged for more than thirty years (Chatterjee et al., 2002). In others study obtained several algorithm for matrix multiplication, but they are not able to beat the running time $\Theta(n^{2.376})$ (Cohn and Umans, 2003; Cohn et al., 2005). The goal of this paper is to draw the attention of theoretical research on what algorithmic calculations does matter most in practice. So, in this situation continuous discussion and culture is heavily needed for the upcoming days to solve a complex problem that takes least time. In a recent study, five recursive layouts have been combined with three parallel matrix multiplication algorithms to show better performance in spite of the complexity overheads of address computation (Chatterjee et al., 2002). Others research have provided a single and multiple precision Graphics Processing Unit implementation of Strassen's and Winograd's algorithm where they have shown that speed up of matrix multiplication process most of the time depends on the volume of the matrices considered for multiplication (Li et al., 2011). Based on a recent study have introduced a technique to multiply matrices which is almost similar to Winograd's method but the time taken to perform this was reduced to $\Theta(n^{2.373})$ by applying tensor power A^n for an initial algorithm A (Williams, 2014). But the drawback of the technique is that it can be applied only to algorithms which satisfy some basic requirements mentioned by them where group theoretic technique is mandatory. In 1987, by using Salem-Spencer theorem, researchers have discussed a technique where the loopholes like dependency on hashing and counting arguments of the previous algorithms have been overcome and Strassen's algorithm has been modified here by replacing the block product of 3 by 2 (Coppersmith and Winograd, 1990). When all the products are of similar shapes, then T-theorem has been able to remove the previous inefficiencies and the resultant matrix exponent has become 2.376. A recent study proposed a method which decreases the complexity of algorithm by Winograd to $(5n^{2.81} + 0.5n^{2.59} + 2n^{2.32} - 6.5n^2)$ for $n = 2^k$ (Cenk and Hasan, 2017). The total arithmetic complexity can be enhanced to $(3.55n^{2.81} + 0.148n^{2.59} + 1.02n^{2.32} - 6.5n^2)$ for $n = 8 \cdot 2^k$, which progresses the best-known bound for a Strassen-like matrix multiplication algorithm. In others research, have shown that the selection of efficient algorithm is dependent on the matrix size as well as the shape (Benson and Ballard, 2015).

A code creation tool has been designed for automatic implementation of multiple sequential and shared-memory parallel variants of every efficient process, including the scheme of parallelization proposed by them. Fast benchmark of 20 fast processes on different problem sizes has been introduced here. Further discussion of real-world implementation matters for these processes on shared-memory machines which is able to precede additional research on creating fast algorithms suitable for real world. A recent study have tried to outperform the existing methods of small matrix multiplication and demonstrated the use of their results by creating multiplication codes over different rings, like integers, polynomials, differential operators and linear recurrence operators (Drevet et al., 2011). In others study by their experimental result, there have shown that Strassen consumes fewer time than row by row and the parallel Strassen is the best one in performance (43% faster than parallel row by row) (Thabet and Ghuribi, 2012). Using the framework of Apache Spark, a fast and scalable technique of matrix multiplication namely STARK has been implemented in that overcome the lacuna in the state of art (Mishra et al., 2018). But the space complexity of the said approach is $\Theta(3^n)$ where the recursion level is denoted by l . The large size matrices consume a large amount of memory whereas the existing techniques run smooth. A Custom Matrix Multiplication (CMM) has been proposed in which outperforms Naïve method and Strassen's method for Discrete Cosine Transform applied to different algorithms for several images in image compression and also it has achieved matrix multiplication for fractional values which was difficult for the existing ones (Rathod et al., 2017).

2. METHODOLOGY

Four methods, namely conventional square matrix multiplication, block matrix multiplication, Strassen's matrix multiplication and Coppersmith - Winograd matrix multiplication have been explained and their

performances have been compared with each other in this section.

2.1 Conventional Square Matrix Multiplication

The conventional square matrix multiplication algorithm is the most fundamental and basic one. Suppose $X = (u_{ij})$ and $Y = (v_{ij})$ are square $(n \times n)$ matrices. Then the product $W = X \times Y$. We define the entry W_{ij} for $i, j = 1, 2, 3, \dots, n$ by

$$W_{ij} = \sum_{k=1}^n u_{ik} \times v_{kj} \text{ Where, } (0 \leq i \leq n \text{ \& } 0 \leq j \leq n)$$

2.1.1 Algorithm

SQUARE_MATRIX_MULTIPLY(X, Y)
 // X, Y and W are $n \times n$ matrices, $W = X \times Y$
Step 1: $n = X.$ rows
Step 2: for $i = 1$ to n
Step 3: for $j = 1$ to n
Step 4: $W_{ij} = 0$
Step 5: for $k = 1$ to n
Step 6: $W_{ij} = W_{ij} + u_{ik} \times v_{kj}$
Step 7: Return W.

In this algorithm, there are three nested for loops i, j and k . Each of for loop runs exactly n times. So, the total time requirement is $\Theta(n^3)$.

2.1.2. Demonstration with Divide & Conquer Technique

When we are going to implement divide & conquer technique for find out the matrix product of $W = X \times Y$, then the value of n must be exact power of 2 in each of the $n \times n$ matrices ($2^1 \times 2^1, 2^2 \times 2^2, 2^3 \times 2^3, 2^4 \times 2^4, \dots, 2^n \times 2^n$) where $n \geq 2$. Here we divide each $n \times n$ matrices into four $(n/2 \times n/2)$ matrices. The dimension $n/2$ is an integer.

Suppose we partition each of X, Y and W matrix into four $(n/2 \times n/2)$ matrices.

$$W = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}_{2 \times 2} \quad X = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}_{2 \times 2} \quad Y = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix}_{2 \times 2}$$

So that we can write,

$$W = X \times Y$$

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}_{2 \times 2} = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}_{2 \times 2} \times \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix}_{2 \times 2}$$

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} = \begin{bmatrix} X_{11}Y_{11} + X_{12}Y_{21} & X_{11}Y_{12} + X_{12}Y_{22} \\ X_{21}Y_{11} + X_{22}Y_{21} & X_{21}Y_{12} + X_{22}Y_{22} \end{bmatrix}$$

Now we can write each element of W matrix as follows-

$$W_{11} = X_{11}Y_{11} + X_{12}Y_{21} \quad (1)$$

$$W_{12} = X_{11}Y_{12} + X_{12}Y_{22} \quad (2)$$

$$W_{21} = X_{21}Y_{11} + X_{22}Y_{21} \quad (3)$$

$$W_{22} = X_{21}Y_{12} + X_{22}Y_{22} \quad (4)$$

Each of the four equations sets out two multiplications of matrices $(n/2 \times n/2)$ and one addition of matrices $(n/2 \times n/2)$. Therefore, total number of multiplications is 8 and total number of additions is 4.

2.1.2 Explanation and Analysis

Now we can implement recursive approach of square matrix multiplication algorithm.

2.1.3.1 SQUARE_MATRIX_MULTIPLY_RECURSIVE(X, Y)

// X, Y and W is $(n \times n)$ matrices. $W = X \times Y$
Step 1: $n = X.$ rows

Step 2: if $n = 1$

Step 3: $W_{11} = X_{11} \times Y_{11}$

Step 4: else partition X , Y and W .

Step 5: Compute W matrix.

$W_{11} = \text{Square\_Matrix\_Multiply\_Recursive}(X_{11}, Y_{11}) +$

$\text{Square\_Matrix\_Multiply\_Recursive}(X_{12}, Y_{21})$

$W_{12} = \text{Square\_Matrix\_Multiply\_Recursive}(X_{11}, Y_{12}) +$

$\text{Square\_Matrix\_Multiply\_Recursive}(X_{12}, Y_{22})$

$W_{21} = \text{Square\_Matrix\_Multiply\_Recursive}(X_{21}, Y_{11})$

$+ \text{Square\_Matrix\_Multiply\_Recursive}(X_{22}, Y_{21})$

$W_{22} = \text{Square\_Matrix\_Multiply\_Recursive}(X_{21}, Y_{12})$

$+ \text{Square\_Matrix\_Multiply\_Recursive}(X_{22}, Y_{22})$

Step 6: Return W .

2.1.3.2 Analysis

We can compute the running time of SQUARE_MATRIX_MULTILY_RECURSIVE ( ) algorithm. Let $T(n)$ is the time requirement for multiplying two $(n \times n)$ matrices.

In the best case, when $n = 1$, we perform just one scalar multiplication in step 3. So, $T(1) = T(1) = \Theta(1)$. The recursive case occurs when $n > 1$; then we partition the matrices in step 4 using index calculations time takes $\Theta(1)$. In step 5, we recursively compute Square_Matrix_Multiply_Recursive ( ) a total of eight times where each recursive computation multiplies one $(n/2 \times n/2)$ matrices. For multiplying each $(n/2 \times n/2)$ matrix time takes $T(n/2)$. So for eight $(n/2 \times n/2)$ matrix multiplication time takes $8T(n/2)$. Step 5 also computes four matrix additions that take constant time $\Theta(n^2)$. Finally, we have computed the resultant matrix W using index calculations to store the result of the matrix additions. Time takes $\Theta(1)$ per entry. Now the overall running time $T(n) = \Theta(1) + 8T(n/2) + \Theta(n^2) + \Theta(1) = 8T(n/2) + \Theta(n^2)$.

$$T(n) = \begin{cases} \Theta(1) & \text{when } (n = 1) \\ 8T(n/2) + \Theta(n^2) & \text{when } (n > 1) \end{cases}$$

Applying Master Theorem, $T(n) = \Theta(n^3)$.

2.2 Block Matrix Multiplication

A block matrix produces several smaller sub matrices, where each block treated as a new matrix.

$$\text{Let, } X = \begin{bmatrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{21} & X_{22} & X_{23} & X_{24} \\ X_{31} & X_{32} & X_{33} & X_{34} \\ X_{41} & X_{42} & X_{43} & X_{44} \end{bmatrix}$$

Now X can be decomposed into four (2×2) sub matrices.

$$A_x = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \quad A_y = \begin{bmatrix} A_{13} & A_{14} \\ A_{23} & A_{24} \end{bmatrix} \quad A_z = \begin{bmatrix} A_{31} & A_{32} \\ A_{41} & A_{42} \end{bmatrix} \quad A_m = \begin{bmatrix} A_{33} & A_{34} \\ A_{43} & A_{44} \end{bmatrix}$$

So that final resultant matrix $X = \begin{bmatrix} X_x & X_y \\ X_z & X_m \end{bmatrix}$

The computation principle of X follows conventional matrix multiplication rules.

2.3 Strassen's Matrix Multiplication

In 1969, German scientist Volker Strassen [3] stunned world of the investigation of his $\Theta(n^{2.807})$ algorithm for the multiplication of square matrices. For carrying out complex linear algebraic operations this algorithm improves the performances of serial processing system.

Construction of Strassen's algorithm follows block matrix multiplication and divides and conquers technique. It is slightly faster than conventional matrix multiplication algorithm where it reduces the number of arithmetic operations. It requires 7 multiplications and 18 addition-subtractions.

With relatively large matrices, the Strassen's algorithm works more efficiently where problem size $n \geq 2$. Now partition of two input matrices X and Y into four equal size blocks as described in block matrix multiplication, and then the algorithm carries out as follows

$$X = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \quad Y = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{11} & Y_{22} \end{bmatrix} \quad W = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}$$

Here, $W = X \times Y$

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \times \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix}$$

2.3.1 Algorithm

STRASSEN_MULTIPLICATION (X, Y)

// X , Y and W is $(n \times n)$ matrices. $W = X \times Y$

Step 1: $n = X$ rows

Step 2: if $n = 1$

Step 3: $W_{11} = X_{11} \times Y_{11}$

Step 4: else partition X , Y and W .

Step 5: Compute M_1 to M_{10} intermediate matrices.

$M_1 = Y_{12} - Y_{22}$, $M_2 = X_{11} + X_{12}$

$M_3 = X_{21} + X_{22}$, $M_4 = Y_{21} - Y_{11}$

$M_5 = X_{11} + X_{22}$, $M_6 = Y_{11} + Y_{22}$

$M_7 = X_{12} - X_{22}$, $M_8 = Y_{21} + Y_{22}$

$M_9 = X_{11} - X_{21}$, $M_{10} = Y_{11} + Y_{12}$

Step 6: Compute Z_1 to Z_7 matrix product.

$Z_1 = X_{11} \times M_1$, $Z_2 = M_2 \times Y_{22}$

$Z_3 = M_3 \times Y_{11}$, $Z_4 = X_{22} \times M_4$

$Z_5 = M_5 \times M_6$, $Z_6 = M_7 \times M_8$

$Z_7 = M_9 \times M_{10}$

Step 7: Compute 4 sub matrices.

$W_{11} = Z_5 + Z_4 - Z_2 + Z_6$, $W_{12} = Z_1 + Z_2$

$W_{21} = Z_3 + Z_4$, $W_{22} = Z_5 + Z_1 - Z_3 - Z_7$

Step 8: Return W .

2.3.2 Analysis

2.4 Coppersmith-Winograd's Matrix Multiplication

In 1989, Coppersmith and Winograd [9] improved Strassen's technique in respect of arithmetic operations. It requires 7 multiplication and 15 addition-subtractions; but mathematically its running time is $\Theta(n^{2.807})$.

2.4.1 Algorithm

COPPERSMITH-WINOGRAD_MULTIPLICATION (X, Y)

// X , Y and W are $(n \times n)$ matrices. $W = X \times Y$

Step 1: $n = X$ rows

Step 2: if $n = 1$

Step 3: $W_{11} = X_{11} \times Y_{11}$

Step 4: else partition X , Y and W .

Step 5: Compute M_1 to M_8 intermediate matrices.

$M_1 = X_{21} + X_{22}$, $M_2 = M_1 - X_{11}$

$M_3 = X_{11} - X_{21}$, $M_4 = X_{12} - M_2$

$M_5 = Y_{12} - Y_{11}$, $M_6 = Y_{22} - M_5$

$M_7 = Y_{22} - Y_{12}$, $M_8 = M_6 - Y_{21}$

Step 6: Compute Z_1 to Z_7 matrix product.

$Z_1 = M_2 \times M_6$, $Z_2 = X_{11} \times Y_{11}$

$Z_3 = X_{12} \times Y_{21}$, $Z_4 = M_3 \times M_7$

$Z_5 = M_1 \times M_5$, $Z_6 = M_4 \times Y_{22}$

$Z_7 = X_{22} \times M_8$

Step 7: Compute Q_1 and Q_2

$Q_1 = Z_1 + Z_2$, $Q_2 = Q_1 + Z_4$

Step 8: Compute 4 sub matrices.

$W_{11} = Z_2 + Z_3$, $W_{12} = Q_1 + Z_5 + Z_6$

$W_{21} = Q_2 - Z_7$, $W_{22} = Q_2 + Z_5$

Step 9: Return W .

2.4.2 Analysis

The overall time requirement calculation is almost same with Strassen's algorithm, only here in step 7 we compute Q_1 and Q_2 value for addition; each addition takes $\Theta(1)$ time. Now running time is $T(n) = \Theta(1) + \Theta(n^2) + 7T(n/2) + \Theta(1) + \Theta(n^2) + \Theta(1) = 7T(n/2) + \Theta(n^2)$. More specifically $T(n)$ can be defined as $T(n) = \Theta(1)$ when $n = 1$ and $7T(n/2) + \Theta(n^2)$ when $n > 1$. Applying Master theorem, the recurrence has the solutions $T(n) = \Theta(n^{2.807})$.

3. EXPERIMENT AND DISCUSSIONS

Figure 1

Figure 2

Figure 3

Figure 4

All our experimental result shown in figure1, figure2, figure3 and figure4 based on different dataset was implemented by MATLAB 9.3 simulator on a serial processing system, where we showed the matrix multiplication computational time will changes respect of different dimensions of input square matrices. From our investigation clearly we understand that Strassen and Winograd matrix multiplication approach improved its performance in terms of time especially for large size matrices with comparing of conventional matrix multiplication approach. The performance level of Strassen's, Winograd and conventional approach are almost same for exactly (2×2) matrices. We observed that due to the little bit reduction of arithmetic operation, in some cases Winograd performs seemingly better than Strassen's algorithm.

4. CONCLUSION

Throughout our paper we have just demonstrate the growth of running time behaviour and functionalities of different square matrix multiplication algorithm with respect of no. of arithmetic operations and dimension of matrices under serial processing system. But we are not examined the implementation of such algorithms in a parallel processing system where time and space both are also important. The applicability of odd dimensional matrices mainly for Strassen and Winograd have not discussed so far. For both the square matrix multiplication algorithm such issues like – reduction the number of multiplication and addition – subtractions, proper rules for the generation of sub matrices till now confusing. So, in addition to fast matrix multiplication research, future investigation should be focused about the mathematical and experimental analysis of these algorithms in different environment with different parameters.

REFERENCES

- Benson A.R and Ballard G, 2015. A Framework for Practical Parallel Fast Matrix Multiplication. In Proc. of the 20th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, January, 42 – 53.
- Bodrato M, 2010. A Strassen - like Matrix Multiplication Suited for Squaring and Higher Power Computation. In Proceedings of 35th International Symposium on Symbolic and Algebraic Computation (ISSAC), 25 – 28 July, Munich, Germany.
- Cenk M and Hasan M.A, 2017. On the Arithmetic Complexity of Strassen – Like Matrix Multiplications. Journal of Symbolic Computation, ACM Digital Library, 484 – 501.
- Chatterjee S., Lebeck A.R, Patnala P.K and Thottethodi M., 2002. Recursive Array Layouts and Fast Matrix Multiplication. IEEE Transactions on Parallel and Distributed Systems 13(11), 1105 – 1123.
- Cohn H. and Umans C., 2003. A Group – Theoretic Approach to Fast Matrix Multiplication. In Proc.FOCS, 44, 438 – 449.
- Cohn H., Kleinberg R., Szegedy B. and Umans C., 2005. Group – Theoretic Algorithms for Matrix Multiplication. In Proc. FOCS, 46, 379 – 388.
- Coppersmith D. and Winograd S, 1990. Matrix Multiplication via Arithmetic Progression. Journal of Symbolic Computation, ACM Digital Library, 251 – 280.
- Drevet C.E, Islam M.N and Schost E, 2011. Optimization techniques for small matrix multiplication. Theoretical Computer Science, May, Elsevier. DOI: 10.1145/1940475.1940488.
- Huss S, Lederman, Jacobson E.M, Johnson J.R, Tsao A and Turnbull T, 1996. Implementation of Strassen's Algorithm for Matrix Multiplication. Supercomputing' 96: Proceedings of the 1996 ACM/IEEE Conference on Supercomputing, Pittsburgh, USA, 32 -32.
- Li J, Ranka S and Sahni S, 2011. Strassen's Matrix Multiplication on GPUs. IEEE 17th International Conference on Parallel and Distributed Systems, Taiwan, 157 – 164.
- Mathew J, Kumar R.V, 2012. Comparative Study of Strassen's Matrix Multiplication Algorithm. In Proceedings of International Journal of Computer Science and Technology (IJCST), ISSN: 0976 - 8491 (online), ISSN: 2229 – 43333 (print), 3(1).
- Mishra C, Bhattacharya S and Ghosh S.K, 2018. Stark: Fast and Scalable Strassen's Matrix Multiplication using Apache Spark. Accepted in IEEE Transaction on Big Data, Last Revised Nov 22.
- Rathod P.M, Vartak A.B and Kunte N, 2017. Optimizing the Complexity of Matrix Multiplication Algorithm. In the Proc. of International Journal of Engineering Research and Technology (IOERT), ISSN-2278-0181, 5(1).
- Thabet K. and Ghuribi S.A, 2012. Matrix Multiplication Algorithms. In Proc. of International Journal of Computer Science and Network Security (IJCSNS), 12(2).
- Williams V.V, 2014. Multiplying matrices in $\Theta(n^{2.373})$ time. Stanford University, July 1.