Summer Research Proposal
Practical Implementation of Matrix Multiplication Algorithm From SUSP

Khai Dong

February 08 2023

1 Matrices and Matrix Multiplication

A *matrix* is a rectangular array of numbers arranged into rows and columns. A matrix with m rows and n columns has $m \times n$ entries (numbers) where m and n are arbitrary positive integers. An example of a 4×3 matrix with 12 entries is

$$\begin{pmatrix} 1 & -12 & 27 & 3 \\ 12 & -31 & 20 & -13 \\ 5 & -12 & -7 & 8 \\ 12 & -5 & 2 & -9 \end{pmatrix}$$

In linear algebra, matrices provide a compact way to represent and describe linear transformations. As such, matrices are used in a large array of disciplines to model multi-dimensional properties, which allows complex properties of the subject matter to be translated into basic mathematical operations. For example, in machine learning, matrices are used to represent multi-dimensional data which, in turn, are used to turn the training process of the machine learning model into linear or logistic regression. Also, another commonly used real-life application of matrices is to correct for relatively in GPS systems.

One of the complex operations involving matrices is matrix multiplication between two matrices. Matrix multiplication involves calculating the dot products of the first matrix's rows with the second matrix's columns. Matrix multiplication is defined if and only if the number of columns in the first matrix is equal to the number of rows in the second matrix. In more mathematical terms, the product of matrix multiplication of matrix A of size $m \times n_1$ and matrix B of size $n_2 \times p$ is defined if and only if $n_1 = n_2 = n$. The resulting matrix C is of size $m \times p$.

Naively, multiplying matrix A and B is done by calculating dot products between rows and columns. One of such dot products costs n plain number multiplications and $n - 1$ number additions to get the sum of the results of the multiplications; the *runtime* of such operation is $O(n)$. Doing that for every entry of C, the runtime of matrix multiplication comes up to $O(n \times m \times p)$. Considering both matrices in the multiplication to be of size $n \times n$, the runtime is $O(n^3)$ or $\omega = 3$, where ω denotes the runtime of the matrix multiplication. The focus of this research involves reducing the number of arithmetic operations involved in calculating the product of two matrices.

2 Previous Works

In 1969, Strassen derived an improved algorithm that yields a runtime complexity of $O(n^{2.81})$ or $\omega = 2.81$ [1]. Currently, the smallest number of ω is 2.37188; this number was presented in 2022 [2].

Currently, Professor Matthew Anderson and his research students are working on improving the speed of matrix multiplication using a theory proposed by Cohn and Uman in 2005. The theory is that *strong uniquely solvable puzzles* (denoted SUSP) can be used to develop a faster algorithm for

matrix multiplication [3].

Theoretically, a larger SUSP will yield a better matrix multiplication algorithm, thus, decreasing ω. Professor Matthew Anderson has developed an algorithm to search for SUSP and managed to find a large SUSP that yields $\omega = 2.505$ [4]. Since this algorithm does not outperform the existing algorithms, the research continues to search for larger SUSPs to further decrease ω.

Cohn-Umans proposed that matrix multiplication of matrix that can be done faster since the cost of doing calculations in *Wedderburn decomposition* is less than the cost of multiplying the matrices [3]. However, the challenge remains to find the Wedderburn decompositions and the inverse Wedderburn decomposition which are the main rigors of Zachary Dubinsky's (2023) thesis paper [5].

3 Puzzles

Puzzles play a large part in Professor Matthew Anderson's current research. A puzzle is defined to be a set of k rows of the same sizes where every entry is either 1, 2, or 3. Within a puzzle, we can permutate the 1s, 2s, or 3s of rows. Consider the following puzzle:

1	2	3
2	3	1
3	1	2

For this puzzle, if we can first permutate the 1s in the first and the second rows, and then, the 1s in the second and third rows, we obtained:

1	2	3	→		2	3/1	→		2	3/1	→	3	2	1
2	3	1		2/1	3			2	3/1			2	1	3
3	1	2		3	1	2		3/1		2		1	3	2

We can see that by doing this, we create overlapping 3s and 1s. Then, we permutate the 3s in the first and the second row and the 3s in the first and the third row to resolve the overlapping 1s and 3s.

	2	3/1	→		2	3/1	→		3/2	1	→	3	2	1	→	3	2	1	
2	3/1			2/1	3			2	1	3		2	1	3		2	1	3	
3/1		2		3	1	2		3/1		2		1	3	2		1	3	2	

In this puzzle, we can see that permutating the 1s, 2s, and 3s to create a new puzzle is possible. If it is impossible to swap the 1s, 2s, and 3s without creating some overlappings, we call that puzzle a uniquely solvable puzzle (USP). An even rarer type of puzzle is called Strong Uniquely Solvable Puzzle (SUSP). A SUSP, besides being a USP, requires that permutating the puzzle would result in 2 pieces overlapping at the exact same places. One example of a SUSP would be

1	3
2	1

A SUSP allows a matrix to be mapped to a *group algebra* which is an important mathematical object being used in Cohn and Umans' matrix multiplication algorithm. The particular of how the mapping works are in Cohn-Uman paper [3]. This would also be a part of my self-studying over the school year to get ready for summer research.

4 Cohn And Umans' Matrix Multiplication Algorithm

Cohn-Umans[3] proposed a fast polynomial multiplication algorithm which is one of the fundamental parts of Zachary Dubinsky's thesis proposal [6]. Cohn-Umans' approach is broken into steps as follows:

- Step 1: Map A and B to their respective group algebra $\bar{A}$ and $\bar{B}$.
- Step 2: Compute the Wedderburn decompositions of $\bar{A}$ and $\bar{B}$, $\vec{A}$ and $\vec{B}$.
- Step 3: Compute the pointwise product of $\vec{A}$ and $\vec{B}$, $\vec{C} = \vec{A} \cdot \vec{B}$.
- Step 4: Compute the inverse Wedderburn decomposition of $\vec{C}$, group algebra $\bar{C}$.
- Step 5: Map $\bar{C}$ back to matrix C.

Steps 1 and 5 of this process are based on the calculation of Discrete Fourier Transform using Fast Fourier Transform described in chapter 30 of *Introduction to Algorithms* [7], which is also a topic to study in research.

The challenge in these steps is largely calculating the Wedderburn decomposition and the inverse Wedderburn decomposition as the process has never been rigorously formulated. The particular of what Wedderburn decompositions are described in the Wedderburn-Artin Theorem, and understanding the proof of this theorem is a part of my learning process.

5 Research Plan and Summer Schedule

My summer research goal is to pick up Zachary Dubinsky's thesis and continue to improve the process of calculating the Wedderburn decompositions and their inverse. To achieve this, I would have to study group theory [8], representation theory[9], Cohn-Uman papers [3], Wedderburn-Artin Theorem, and Zachary Dubinsky's thesis [5].

Next term, I will take MTH-332 as a part of my math major and continue my work study with Professor Matthew Anderson. MTH-332 should provide me with a minimum understanding of group theory. I will also study representation theory on my own to be able to understand Cohn and Uman's publications and the Wedderburn-Artin theorem and its proof. This gives me time to study more group and representation theory that is directly related to SUSP, and how SUSP can be turned into a matrix multiplication algorithm. his part will provide the fundamental knowledge that I need to continue the research.

The second part would be analyzing the existing prototype Professor Anderson and his research students have created and worked on, and actually doing research on how the code and the algorithm can be further optimized. For this part, I will pick up on Zachary Dubinsky's thesis [5] in implementing Cohn-Umans' algorithm and try to fully implement it. However, the main goal is still reducing the exponential runtime involved in this process.

I will carry out the first part in the remainder of the 2022-2023 school year as part of classes and work-study and do the second part during the summer.
My summer research timeline will be as follows:

Time	Activity	Expected Results
Week 1	Finishing understanding Cohn and Uman's theories	Have an understanding of how a matrix multiplication algorithm is generated from SUSP.
Week 2	Finishing understanding Zachary Dubinsky's thesis on calculating Wedderburn decompositions	Have a understanding of how representation and group theory are involved in the process of generating a matrix multiplication algorithm.
Week 3	Analyzing the current prototype implementation by Professor Matthew Anderson and Zachary Dubinsky of Cohn and Umans' algorithm	Understand implementation and identify where the implementation can be improved.
Week 4	Continuing analyzing the prototype implementation and studying the related topics on how to further improve the algorithm runtime	Continue to understand the implementation and how it could be improved.
Week 5	Try to improve the implementation	Implement the improvement, and run tests to see if the overall runtime improved.
Week 6	Continuing improve the implementation	General code optimization
Week 7	Try to minimize RAM and CPU usage to further optimize the runtime, and possibly introduce parallelism	Minimize RAM and CPU usage, and strategize how to introduce parallelism.
Week 8	Implement synchronization code if possible, test the code	Test the final version. Compare the result with the prototype.

References

[1] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969. URL http://eudml.org/doc/131927.

[2] Ran Duan, Hongxun Wu, and Renfei Zhou. Faster matrix multiplication via asymmetric hashing. 2022. doi: 10.48550/ARXIV.2210.10173. URL https://arxiv.org/abs/2210.10173.

[3] H. Cohn, R. Kleinberg, B. Szegedy, and C. Umans. Group-theoretic algorithms for matrix multiplication. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 379–388, 2005. doi: 10.1109/SFCS.2005.39.

[4] Matthew Anderson, Zongliang Ji, and Anthony Yang Xu. Matrix multiplication: Verifying strong uniquely solvable puzzles, 2023. URL https://arxiv.org/abs/2301.00074.

[5] Zach Dubinsky. Computing wedderburn decompositions. 2023.

[6] Zach Dubinsky. Is fast matrix multiplication possible? 2023.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. ISBN 0262033844.

[8] David S Dummit and Richard M Foote. *Abstract Algebra*. John Wiley & Sons, Nashville, TN, 3 edition, June 2003.

[9] William Fulton and Joe Harris. *Representation theory*. Graduate Texts in Mathematics. Springer, New York, NY, 1 edition, January 1991.