Cohn-Umans Matrix Multiplication

Khai Dong   Matthew Anderson (Advisor)

Computer Science Department, Union College

Abstract

Cohn and Umans [3] proposed a theoretical framework for faster matrix multiplication and conjectured an upper bound of $\mathcal{O}(n^{2+o(1)})$ on its runtime. This research aims to implement this algorithm and verified its runtime.

Introduction

Matrices are used in a large array of disciplines including mathematics and computer science. Improving the runtime of the operation is of great significance. Naive multiplication of $n \times n$ matrices involves taking the dot product of every row of the left matrix with every column of the right matrix. This uses $3n^3$ additions and multiplications of scalar numbers. In other terms, this algorithm is in $\mathcal{O}(n^3)$ or has $\omega = 3$.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a \cdot e + b \cdot g & a \cdot f + b \cdot h \\ c \cdot e + d \cdot g & c \cdot f + d \cdot h \end{bmatrix}$$

Figure 1: Multiplying 2×2 matrices.

Multiple improvements have been made over the years:

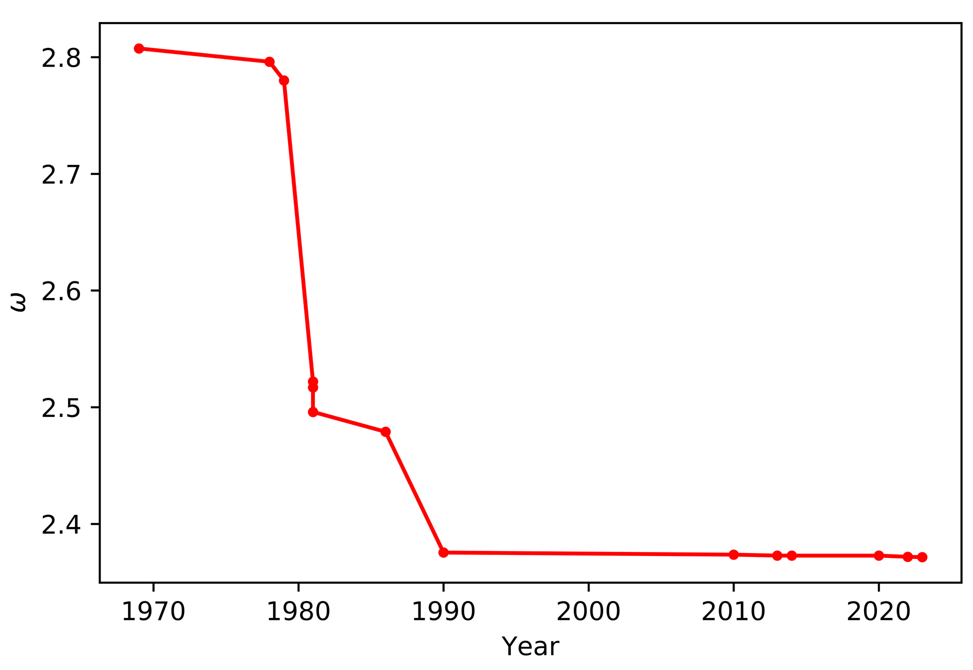

Figure 2: Improvement of estimates of exponent ω

Cohn and Umans [3] proposed a framework that conjectured $\omega = 2 + o(1)$ and showed $\omega \leq 2.41$ [2]. The framework is largely unexplored and unimplemented.

Question

How to multiply matrices using Cohn-Umans framework? Can we determine what is ω for a given SUSP? When do we get improvements over the naive one?

We have successfully implemented the computation process for computing the Wedderburn Decomposition and Cohn-Umans matrix multiplication is done. This implementation is done in **SageMath** 10.0.

Strassen's Matrix Multiplication

The improvement given by Cohn-Umans' matrix multiplication relies on similar concept to Strassen's [4]: doing 2×2 matrix multiplication using 7 multiplications instead of 8 and $\mathcal{O}(n^2)$ additions.

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \cdot \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \to \begin{bmatrix} P_1 & & \\ & \dots & \\ & & P_7 \end{bmatrix}$$

$$P_1 = (A_{11} + A_{22}) \cdot (B_{11} + B_{22}),$$
$$P_2 = (A_{21} + A_{22}) \cdot B_{11}, \quad P_5 = (A_{11} + A_{12}) \cdot B_{22},$$
$$P_3 = A_{11} \cdot (B_{12} - B_{22}), \quad P_6 = (A_{21} - A_{11}) \cdot (B_{11} + B_{12}),$$
$$P_4 = A_{22} \cdot (B_{21} - B11), \quad P_7 = (A_{12} - A_{22}) \cdot (B_{21} + B_{22}).$$

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} P_1 + P_4 - P_5 + P_7 & P_3 + P_5 \\ P_2 + P_4 & P_1 - P_2 + P_3 + P_6 \end{bmatrix}$$

Figure 3: Strassen's Matrix Multiplication.

This gives $\omega = 2.81$. If matrix multiplication of $m \times n$ and $n \times r$ matrices is possible in less than mnr multiplications and $\mathcal{O}(n^2)$ additions, we gain better performance.

Cohn-Umans' Matrix Multiplication

This framework relies on a **Strong Unique Solvable Puzzle** and an integer p to generate a **group algebra** which realizes multiplication of $m \times n$ and $n \times r$ matrices.

$$\begin{tikzcd} A, B \arrow[r, "\text{Step 1}"] \arrow[d, dashed] & \bar{A}, \bar{B} \arrow[r, "\text{Step 2}"] \arrow[d, dashed] & \Phi(\bar{A}), \Phi(\bar{B}) \arrow[d, "\text{Step 3}"] \\ C & \bar{C} \arrow[l, "\text{Step 5}"'] & \Phi(\bar{C}) \arrow[l, "\text{Step 4}"'] \end{tikzcd}$$

Figure 4: Cohn and Umans Matrix Multiplication

Steps 1 and 5 are similar to **Fast Fourier Transform** where we embed the matrices A and B into the **group algebra**. Steps 2 and 4 involve applying and inversing **Wedderburn Decomposition** Φ on the embedded matrices. Step 3 is just multiplying the sets of k matrices.

In this framework, m, n, r, and k are determined by the **group algebra** which is determined by the SUSP and p. If k is sufficiently small compared to mnr, we get efficiency improvement compared the the naive method.

Computing Φ is a nontrivial task.

Implementation

One problem in our implementation is the error propagation from continuously applying and inversing Φ.

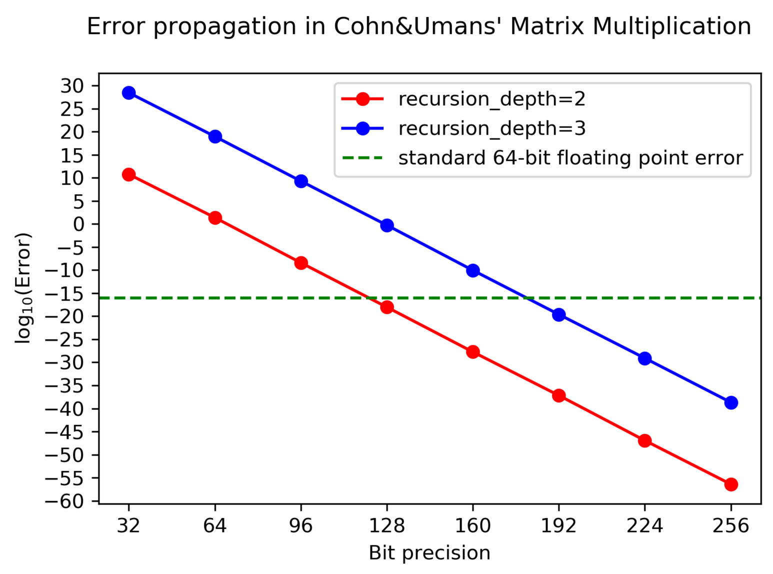

Figure 5: Cohn-Umans' Matrix Multiplication's Error propagation

However, approximating Φ is the only solution we have to obtain Φ in reasonable time.

Asymptotic Bounds

We have been able to derive an upperbound for ω in terms of the puzzle parameters, and some experiments to test our bound against Cohn&Umans [2] using puzzle

$$P = \begin{array}{|c|c|} \hline 1 & 3 \\ \hline 2 & 1 \\ \hline \end{array}$$

Our ω-upperbounds are not as good as Cohn-Umans [2]; However, the 3 upperbounds for ω converge towards each others as p increases.

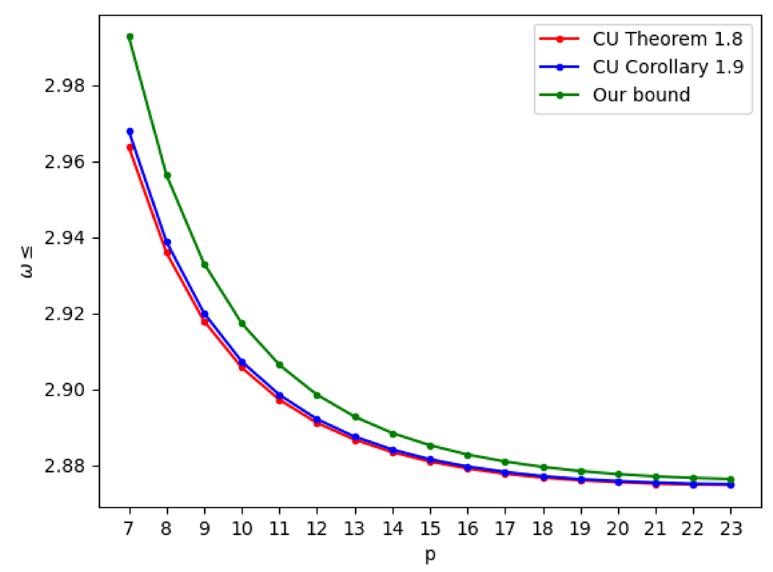

Figure 6: Comparisons of ω against Cohn&Umans'

Thresholds of Matrices' Sizes

We determine the threshold for the matrices' sizes where Cohn-Umans' algorithm gives improvement, n_0. Table 1 shows the n_0's for SUSP P.

p	$n_0 \approx$
8	$2.013 \cdot 10^8$
9	$5.165 \cdot 10^8$
10	$1.200 \cdot 10^9$
11	$2.572 \cdot 10^9$

Table 1: Some n_0's for puzzle P

These values for n_0 are big for the usual application of matrix multiplication which impedes our efforts to measure the actual runtime.

Runtime Simulation

Efforts have been put into simulating the number of operations used instead. However, due the time constraints and the fact that meaningful data only appear at $n >> n_0$, meaningful attempts to analyze the runtime has yet to be carried out.

Acknowledgments

Thanks to Davenport Summer Research Fellowship for sponsoring this research, to Zachary Dubinsky for the helpful discussions and materials, and to my colleagues for the continuous and helpful support.

References

[1] Anderson, M., Ji, Z., and Xu, A. *Matrix Multiplication: Verifying Strong Uniquely Solvable Puzzles.* 06 2020, pp. 464–480.

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

[3] Cohn, H., and Umans, C. A group-theoretic approach to fast matrix multiplication. *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.* (2003), 438–449.

[4] Strassen, V. Gaussian elimination is not optimal. *Numerische Mathematik 13* (1969), 354–356.