Group Theoretic Matrix Multiplication Optimization

UNION COLLEGE

Jingyu Yao, Matthew Anderson

CS

Background

Matrix Multiplication is one of the most fundamental yet important mathematical operations across a variety of of subjects, including physics, engineering, and computer science. Traditional matrix multiplication uses a computational complexity of cube operations with respect to the size of the matrix, n, denoted $O(n^3)$.

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{bmatrix}$$

Figure 1. Traditional matrix multiplication. For this 3 by 3 matrix, since $C_1 = a_1b_1 + a_2b_4 + a_3b_7$, and we must compute 9 such *Cs*, we require a total of $3^3 = 27$ operations.

Our Approach

Previously, Prof. Anderson and his students have developed an algorithm using Uniquely Solvable Puzzles(USP) to generate groups to accelerate the computational process of matrix multiplication[1]. A USP is a puzzle that it is impossible to create a non-overlapping new puzzle by switching the rows of the numbers 1, 2, or 3 while maintaining their respective positions.

Compared to the traditional $O(n^3)$ computational complexity, this algorithm can reduce the computational complexity of matrix multiplication to below $O(n^3)$[2].

1	1	1
3	2	1
3	3	2

Figure 2. A USP.

1	2	3
2	3	1
3	1	2

Not a USP.

Larger USPs can generate better groups that can yield an efficient algorithm for matrix multiplication, with a theoretic best complexity of $O(n^{2.41})$ [2].

The Intuition Behind Using USPs

The intuition behind this group theoretic algorithm comes from the known method of using Discrete Fourier Transform(DFT) to compute polynomial multiplication.

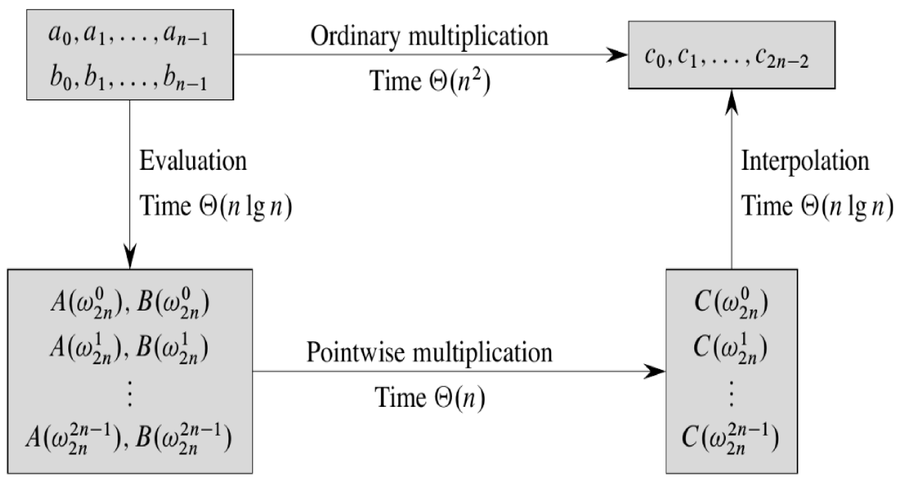

Figure 3[3]. Polynomial multiplication using DFT. Since the computation complexity of performing a Fourier transform, including both evaluating into Fourier domain and interpolating back to normal matrix, both takes $O(n \cdot \lg n)$, this algorithm can asymptotically reduce computational complexity below $O(n^2)$.

Algorithm Optimization

In this research, we transform matrix multiplication into group algebra using USPs, and the computational complexity of group algebra can be reduced below $O(n^3)$.

While Prof. Anderson previously created a working implementation based on this algorithm, this summer I focused on optimizing his algorithm by adopting better programming practices and incorporating standard C++ libraries. The updated implementation of the algorithm transformed all structs in C into classes in C++, which provides better modularity and information hiding. The updated implementation also incorporated standard C++ libraries as much as possible, which leads to better performance and compatibility across different platforms.

Future Work

Our implementation in the multiplication operation for group algebra can be further improved using representation theory and recursion[2]. In particular, multiplication for group algebra and be represented as *block-diagonal matrix multiplication,* which would allow us to perform recursion.

$$A = \begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_q \end{bmatrix} \quad B = \begin{bmatrix} B_1 & 0 & \cdots & 0 \\ 0 & B_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B_q \end{bmatrix}$$

Figure 4. Group multiplication in the group algebra $\mathbb{C}[G]$ can be represented as *block-diagonal matrix multiplication,* where the dimension of these blocks are the dimensions of the irreducible representations of G.

We plan to utilize external libraries to handle the transformation from groups to the irreducible representations.

```
gap> g:= AlternatingGroup( 4 );;
gap> repr:= IrreducibleRepresentations( g );
[ Pcgs([ (2,4,3), (1,3)(2,4), (1,2)(3,4) ]) ->
    [ [ [ 1 ] ], [ [ 1 ] ], [ [ 1 ] ] ],
  Pcgs([ (2,4,3), (1,3)(2,4), (1,2)(3,4) ]) ->
    [ [ [ E(3) ] ], [ [ 1 ] ], [ [ 1 ] ] ],
  Pcgs([ (2,4,3), (1,3)(2,4), (1,2)(3,4) ]) ->
    [ [ [ E(3)^2 ] ], [ [ 1 ] ], [ [ 1 ] ] ],
  Pcgs([ (2,4,3), (1,3)(2,4), (1,2)(3,4) ]) ->
    [ [ [ 0, 0, 1 ], [ 1, 0, 0 ], [ 0, 1, 0 ] ],
      [ [ -1, 0, 0 ], [ 0, 1, 0 ], [ 0, 0, -1 ] ],
      [ [ 1, 0, 0 ], [ 0, -1, 0 ], [ 0, 0, -1 ] ] ] ]
```

Figure 5. Examples of using GAP (gap-system.org) to generate irreducible representations.

Acknowledgements

Thanks to Union College and the department of Undergraduate Research for hosting and supporting this summer research.

References

[1]Anderson, Matthew, Zongliang Ji, and Anthony Yang Xu. "Matrix Multiplication: Verifying Strong Uniquely Solvable Puzzles." *International Conference on Theory and Applications of Satisfiability Testing*. Springer, Cham, 2020.

[2]Cohn, Henry, et al. "Group-theoretic algorithms for matrix multiplication." *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*. IEEE, 2005.

[3]Cormen, Thomas H, and Thomas H. Cormen. *Introduction to Algorithms*. Cambridge, Mass: MIT Press, 2001. Print.