Research Statement – Josh Alman

I am a theoretical computer scientist. I work in both algorithm design and complexity theory, and much of my research combines tools and insights from both areas. I have a wide-ranging interest in these areas: I have designed algorithms for many problems including nearest neighbor search [AW15, ACW16, ACW20], processing data streams [AY20], high-dimensional statistics [Alm19a, ACSS20], matchmaking in multi-agent systems [AM17], public-key cryptography [AH19] and dynamic graph problems [AMW17], and I have proved results in many areas of complexity theory including fine-grained complexity [AM17, AMW17, AVW20], circuit complexity [ACW16, AW17, AC19], communication complexity [AWY18, AC19], and data structure lower bounds [AMW17, AWY18].

In addition to the above areas, one particular theme which I have studied extensively is **algorithms for fundamental algebraic problems**, and their applications. Algorithms for algebraic problems like matrix multiplication, solving linear systems and linear programs, and computing Fourier transforms, have many known uses in nearly every area of computer science. I'm particularly interested in finding ways to apply algebraic tools in new settings. For example, in my recent work with Huacheng Yu [AY20], we used a new connection with matrix multiplication to speed up sketching algorithms for processing data streams. This approach hadn't been used in streaming algorithms before, and it allowed us to speed up algorithms for important, practical problems which hadn't been improved in 15 or more years. Improving algebraic tools, and understanding and employing their versatility, is imperative to the theory and practice of computation.

In the remainder of this statement, I will focus on my research on three topics related to this theme:

1. Algorithms for **matrix multiplication**. With Virginia Vassilevska Williams, I designed the fastest algorithm to date, and explained limitations of the known techniques which prevent us from designing even faster, optimal algorithms.
2. Algorithms for computing important **linear transforms** such as Fourier transforms. My work in this area focuses on a technique called **matrix rigidity**, which analyzes how well the matrices underlying these transforms can be decomposed as the sum of a sparse matrix and a low-rank matrix. With Ryan Williams, I found a new, efficient decomposition for the Walsh-Hadamard transform, and with Lijie Chen, I gave the first construction of a rigid matrix which doesn't have an efficient decomposition.
3. **Novel applications** of these algebraic tools to solve diverse problems throughout computer science.

1 Matrix Multiplication Algorithms

Matrix multiplication is one of the most basic algebraic operations. Many other fundamental computational tasks involving matrices, including solving linear systems and linear programs, and computing the inverse or determinant of a matrix, can be performed in the same amount of time as it takes to multiply matrices. Since the surprising breakthrough algorithm of Strassen [Str69] which showed that matrices can be multiplied faster than the most straightforward algorithm, algorithmic problems from nearly every area of computer science have been sped up by clever reductions to matrix multiplication.

Despite the importance of matrix multiplication, there is still a big gap between our best algorithms and the what we conjecture to be possible: It is popularly conjectured that two $n \times n$ matrices can be multiplied in roughly $O(n^2)$ time[1], but we just don't know the algorithmic techniques needed to achieve this. My research on matrix multiplication has consisted of a two-pronged attack: I have designed the fastest known algorithm for matrix multiplication, which runs in time $O(n^{2.37286})$, and I have formally explained why a vast generalization of the current techniques cannot come close to an $O(n^2)$ time algorithm.

[1]Matrices with large numbers as entires take longer to multiply; in this section I assume that the entries are small enough that this doesn't play a substantial role in the final running time, although the algorithms I discuss here extend to that setting as well.

My new fastest algorithm for matrix multiplication is very recent joint work with one of my PhD advisors, Virginia Vassilevska Williams [AV21]. We give the first improved running time for matrix multiplication in 7 years. Our algorithm builds on the seminal work of Coppersmith and Winograd [CW87] from 1987. Since then, the few other improvements [Vas12, DS13, LG14] have come from modifying Coppersmith and Winograd's algorithm in a methodical way (by taking 'Kronecker powers'). We take a different approach: we give a new and improved way to *analyze* matrix multiplication algorithms, and applying our analysis to essentially the same algorithm from past work gives our better running time. At a high level, we find that quite a bit of additional information can be computed from the results of intermediate steps of the best matrix multiplication algorithms, and some of that information is helpful for speeding up the algorithm.

In another recent line of work [AV18a, AV18b, Alm19b], we focus on *ruling out* approaches to designing an $O(n^2)$ time algorithm. We show that vast generalizations of the known approaches to designing matrix multiplication algorithms cannot achieve running time $O(n^2)$, or even $O(n^{2.15})$. In other words, in order to get close to an $O(n^2)$ time algorithm, which is widely believed to be possible, an entirely new approach is needed. My work [Alm19b], which gave the most general limitation result, received the **best student paper award at CCC 2019**.

This complexity-theoretic project of proving barriers to matrix multiplication algorithms may at first seem like a negative result, but it was an important inspiration for our new algorithm. To help design our algorithm, we pinpointed aspects of our barrier where there is a gap between our lower bound and the running time that known algorithms could achieve, and focused on finding new algorithmic ideas to close the gap. This is one way in which my work uses a synergy between complexity theory and algorithm design.

2 Linear Transforms and Matrix Rigidity

Some of the most important and applicable algorithms are for computing linear transforms such as the Fast Fourier transform and the Walsh-Hadamard transform. Much of my work studying these transforms focuses on a technique called *matrix rigidity*. A matrix is called rigid if it cannot be written as the sum of a low-rank matrix and a sparse matrix. A long line of work initiated by Leslie Valiant [Val77] has shown that determining the rigidity of the matrix underlying a linear transform can be the key to understanding that transform: if the matrix is rigid, then lower bounds in many models of computation are known to hold, and if the matrix is not rigid, then many efficient algorithms and small circuits follow.

Even though rigidity was defined over 40 years ago and has numerous applications, it is still poorly understood: we don't know how rigid almost any particular matrix is, or how to show that *any* explicit matrix (like a Fourier transform) is rigid. My research has addressed this challenge by showing that previous leading candidates for rigid matrices are *not* rigid, and by giving the first nontrivial construction of rigid matrices.

With my other PhD advisor, Ryan Williams [AW17], we focus on the rigidity of the Walsh-Hadamard transform. This transform had long been conjectured to be rigid, and many prior papers proved partial rigidity results for it. In our work, we prove that the Walsh-Hadamard transform is *not* rigid, by giving a surprising new way to express it as the sum of a low-rank matrix and a sparse matrix. In addition to explaining the lack of success of the prior work on proving this transform is rigid, our construction also leads to new algorithms and circuits for the Walsh-Hadamard transform, including smaller low-depth circuits which can be evaluated more efficiently in parallel.

With Lijie Chen [AC19], we give the first nontrivial construction of rigid matrices. By using the many known connections between matrix rigidity and other models of computation, our construction has a number of implications in complexity theory. For instance, using a connection with communication complexity due to Razborov [Raz89], our construction gives the first lower bound against the communication complexity-analogue of the 'Polynomial Hierarchy'. The new approach we introduce for constructing complex objects like rigid matrices has already been used by other complexity theorists [BHPT20, Vio20]. Our work received the **best student paper award at FOCS 2019**.

3 Applications Throughout Computer Science

In addition to developing algebraic tools in computer science, I broadly work in many areas of algorithms and complexity theory. A few of my projects in these areas involve finding new ways to apply algebraic tools to problems where they hadn't been used before. My favorite results along these lines include:

- Faster algorithms for batch nearest neighbor search, where one is given sets of data points and query points, and one wants to find the most similar data point to each query point, according to some distance measure [AW15, ACW16, ACW20]. We give the first subquadratic time algorithm for the exact problem in high dimensions, and the fastest known algorithm for the approximate problem, for various distance measures including Hamming and Euclidean distance.
- Faster algorithms for maintaining linear sketches like CountSketch and CountMinSketch, which allow one to efficiently maintain estimates of many statistics about data streams [AY20]. Our algorithm gives the first speedup for maintaining these sketches since they were introduced over 15 years ago.
- A data structure lower bound, showing that the known data structures for maintaining which nodes are connected to each other in a graph whose edge set can undergo changes (insertions and deletions) are already the best possible [AWY18], even when the data structure is randomized and may err on 49.99% of queries (simply outputting coin flips would achieve 50% trivially).

4 What's Next?

I plan to continue to develop algebraic tools in computer science like algorithms for matrix multiplication and linear transforms, to continue to expand on the connections between these algebraic techniques and other diverse areas of computer science, and to more broadly attack many problems throughout algorithms and complexity. For some examples of questions along these lines that I aim to answer:

1. Can we design a *faster* Fourier transform algorithm? Either designing a faster algorithm, or proving that one cannot exist, would have tremendous impact. I am optimistic that I can make real progress on this difficult problem because my work gives techniques for circumventing two major barriers. First, Dvir and Liu [DL19] recently extended my and Williams' non-ridigity result [AW17] to show that Fourier transforms are not rigid, which is a necessary step toward designing a substantially faster algorithm. Second, it's known that algebraic circuits with 'bounded coefficients' cannot be used to speed up the fast Fourier transform. I've recently found the first construction of smaller circuits with 'unbounded coefficients' for other transforms including Hadamard transforms, which I hope to extend to the Fourier transform.

2. Can we bridge the gap between *theoretical* and *practical* matrix multiplication algorithms? The best recent theoretical algorithms involve large constant factors. This doesn't mean that matrix multiplication is impractical: much *engineering* work has gone into designing fast practical algorithms, especially using a massively parallel approach, which are leveraged for speeding up other algorithms. That said, incorporating more of the known theoretical advances could have an extraordinary impact. I plan to extend my work on matrix multiplication barriers to more carefully analyze the constant factors involved, with the goal of finding which techniques could be used in a more practical setting.

3. Can we prove new barrier results for important algorithmic problems, or circumvent more known barriers? A barrier result shows that known algorithmic techniques for a problem cannot yield a faster algorithm. Algorithm designers often interpret this as suggesting that further improvements for the problem are unlikely. Much of my work instead uses barriers to pinpoint new algorithmic techniques to help solve the problem. As I discussed earlier, this approach helped lead to my new matrix multiplication algorithm. It also motivated my and Yu's new streaming algorithms [AY20]: it was known that no 'nonadaptive' algorithm could substantially speed up the state of the art, but we found how to use matrix multiplication to design an 'adaptive' algorithm. I plan to use this approach for other problems where barriers against nonadaptive algorithms are known. More generally, I plan to use barriers to drive further research in both algorithm design and complexity theory; the two paragraphs above both follow this plan of attack.

References

[AC19] Josh Alman and Lijie Chen. Efficient construction of rigid matrices using an np oracle. In *FOCS*, 2019.

[ACSS20] Josh Alman, Timothy Chu, Aaron Schild, and Zhao Song. Algorithms and hardness for linear algebra on geometric graphs. In *FOCS*, 2020.

[ACW16] Josh Alman, Timothy M. Chan, and Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *FOCS*, 2016.

[ACW20] Josh Alman, Timothy M. Chan, and Ryan Williams. Faster deterministic and las vegas algorithms for offline approximate nearest neighbors in high dimensions. In *SODA*, 2020.

[AH19] Josh Alman and Robin Hui. Predicate encryption from bilinear maps and one-sided probabilistic rank. In *TCC*, 2019.

[Alm19a] Josh Alman. An illuminating algorithm for the light bulb problem. In *SOSA*, 2019.

[Alm19b] Josh Alman. Limits on the universal method for matrix multiplication. In *CCC*, 2019.

[AM17] Josh Alman and Dylan McKay. Theoretical foundations of team matchmaking. In *AAMAS*, 2017.

[AMW17] Josh Alman, Matthias Mnich, and Virginia Vassilevska Williams. Dynamic parameterized problems and algorithms. In *ICALP*, 2017.

[AV18a] Josh Alman and Virginia Vassilevska Williams. Further limitations of the known approaches for matrix multiplication. In *ITCS*, 2018.

[AV18b] Josh Alman and Virginia Vassilevska Williams. Limits on all known (and some unknown) approaches to matrix multiplication. In *FOCS*, 2018.

[AV21] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *SODA*, 2021.

[AVW20] Josh Alman and Virginia Vassilevska Williams. Ov graphs are (probably) hard instances. In *ITCS*, 2020.

[AW15] Josh Alman and Ryan Williams. Probabilistic polynomials and hamming nearest neighbors. In *FOCS*, 2015.

[AW17] Josh Alman and Ryan Williams. Probabilistic rank and matrix rigidity. In *STOC*, 2017.

[AWY18] Josh Alman, Joshua R Wang, and Huacheng Yu. Cell-probe lower bounds from online communication complexity. In *STOC*, 2018.

[AY20] Josh Alman and Huacheng Yu. Faster update time for turnstile streaming algorithms. In *SODA*, 2020.

[BHPT20] Amey Bhangale, Prahladh Harsha, Orr Paradise, and Avishay Tal. Rigid matrices from rectangular pcps. In *FOCS*, 2020.

[CW87] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. In *STOC*, 1987.

[DL19] Zeev Dvir and Allen Liu. Fourier and circulant matrices are not rigid. In *CCC*, 2019.

[DS13] Alexander Munro Davie and Andrew James Stothers. Improved bound for complexity of matrix multiplication. *Proceedings. Section A, Mathematics-The Royal Society of Edinburgh*, 143(2):351, 2013.

[LG14] François Le Gall. Powers of tensors and fast matrix multiplication. In *ISAAC*, 2014.

[Raz89] A. A. Razborov. On rigid matrices (in Russian). 1989.

[Str69] Volker Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.

[Val77] Leslie G Valiant. Graph-theoretic arguments in low-level complexity. In *MFCS*, 1977.

[Vas12] Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *STOC*, 2012.

[Vio20] Emanuele Viola. New lower bounds for probabilistic degree and ac0 with parity gates. In *ECCC*, 2020.