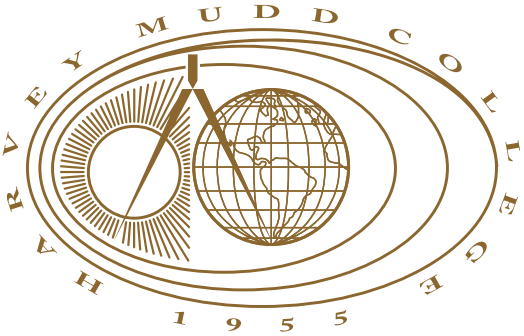
2012

Uniquely Solvable Puzzles and Fast Matrix Multiplication

Palmer Mebane
Harvey Mudd College



Uniquely Solvable Puzzles and Fast Matrix Multiplication

Palmer Mebane

Michael Orrison, Advisor

Nicholas Pippenger, Reader

May, 2012

HARVEY MUDD COLLEGE

Department of Mathematics

Abstract

In 2003 Cohn and Umans introduced a new group-theoretic framework for doing fast matrix multiplications, with several conjectures that would imply the matrix multiplication exponent ω is 2. Their methods have been used to match one of the fastest known algorithms by Coppersmith and Winograd, which runs in $O(n^{2.376})$ time and implies that $\omega \leq 2.376$. This thesis discusses the framework that Cohn and Umans came up with and presents some new results in constructing uniquely solvable puzzles that were introduced in a 2005 follow-up paper, and which play a crucial role in one of the $\omega = 2$ conjectures.

Contents

Abstract **iii**

1 Introduction **1**

2 Group-Theoretic Framework **5**
2.1 Group Algebras 5
2.2 Triple Product Property 12
2.3 Extraneous Group Algebra Computations 15

3 Uniquely Solvable Puzzles **21**
3.1 Definition and Understanding 21
3.2 Application 27
3.3 Current Progress 31
3.4 Left-Right Paradigm 33
3.5 SET Paradigm 38

Bibliography **43**

List of Figures

3.1 Example USP from Cohn et al. (2005). 22
3.2 Pieces of the USP in Figure 3.1. 23
3.3 Pieces of the USP in Figure 3.1 with 3s placed. 24
3.4 Pieces of the USP in Figure 3.1 with 3s and a row of 1s placed. 24
3.5 Pieces of the USP in Figure 3.1 with 3s and 1s placed. 25
3.6 Small USP (left) and two example elements of the group H. . 28
3.7 Example USP using Left-Right Paradigm. 34
3.8 Pieces of the USP in Figure 3.7 with the 3s fixed. 34
3.9 Example cap C of $\mathbb{F}_3^3$ and associated USP $F(C)$ 39
3.10 Example cap C of $\mathbb{F}_3^3$ and associated USP $G(C)$. 41

List of Tables

3.1 Summary of case analysis needed in the proof of Lemma 2. . 41

Chapter 1

Introduction

When studying algorithms for linear algebra, fast matrix multiplication is perhaps the most important problem with respect to both theory and application. Several other tasks can be reduced to some form of matrix multiplication. As summarized in Bürgisser et al. (1997), computing the determinant and characteristic polynomial are some of the problems whose complexities are determined entirely by that of matrix multiplication. Matrix multiplication also plays a role in more hands-on problems such as search engine algorithms. (Brin and Page, 1998)

The naive algorithm for multiplying two $n \times n$ matrices requires summing n terms to compute each of the n^2 entries of the product. This means the algorithm takes a number of steps bounded above by some constant number times n^3, which we write using big-O notation as $O(n^3)$. Matrix multiplication algorithms are often measured in terms of the number of multiplications required, since this is usually a more expensive operation in practice than addition. For the first several decades of algorithms research, no better method than the naive one was known. (Robinson, 2005)

The first improvement on the naive algorithm was published in Strassen (1969). As described in Robinson (2005), to compute the product of two $2n \times 2n$ matrices A, B, Strassen's algorithm split each factor into four $n \times n$ blocks as follows:

$$A = \begin{bmatrix} X & Y \\ Z & W \end{bmatrix}, \qquad B = \begin{bmatrix} X' & Y' \\ Z' & W' \end{bmatrix}.$$

Obtaining the desired product AB requires computing eight $n \times n$ matrix products: XX', XY', ZX', WW', YW', YZ', ZY', WZ'. The ordinary $O(n^3)$ algorithm is equivalent to doing all eight multiplications directly.

Strassen's algorithm however takes advantage of the fact that adding matrices is quicker than multiplying them, and instead computes the following seven products:

$$C_1 = (X+Y)W', \;\; C_2 = (Z+W)X', \;\; C_3 = X(Y'-W'), \;\; C_4 = W(Z'-X'),$$

$$C_5 = (X+W)(X'+W'), \; C_6 = (Z-X)(X'+Y'), \; C_7 = (Y-W)(Z'+W').$$

We can then obtain AB as follows:

$$AB = \begin{bmatrix} -C_1 + C_4 + C_5 + C_7 & C_1 + C_3 \\ C_2 + C_4 & -C_2 + C_3 + C_5 + C_6 \end{bmatrix}.$$

The seven $n \times n$ matrix products can be computed recursively with the same method. A simple running time analysis shows that Strassen's algorithm runs in $O(n^{\log_2 7})$ time, where $\log_2 7 \approx 2.807 < 3$, meaning that this is an improvement over the naive algorithm.

Strassen's ideas and results opened the floodgates for others to attempt to bring the complexity down farther, and the infimum of all possible exponents on matrix multiplication complexity was eventually given the name ω. Coppersmith and Winograd (1990) has one of the most efficient known algorithms with a complexity of $O(n^{2.376})$, implying $\omega \leq 2.376$. The algorithm they give defies a concise explanation, but the basic idea is to construct particular polynomial identities that imply a method for computing linear combinations of a small number of matrix products, similar to the idea of Strassen's algorithm but much more streamlined and intricate. A small improvement on their methods was recently found in Williams (2011), which obtains $\omega < 2.3727$, the current best known upper bound.

The best known lower bound for ω is 2, a trivial bound obtained by noticing that there are n^2 entries that have to be computed in the product matrix. Many algorithms researchers believe this bound can be achieved, as suggested by the multiple conjectures implying $\omega = 2$ in Cohn et al. (2005) and Coppersmith and Winograd (1990).

Cohn and Umans (2003) presented a new framework for matrix multiplication based on finite groups and representation theory. Given a finite group G and two elements in the associated group algebra $\mathbb{C}[G]$, methods exist for performing the multiplication of two elements $\alpha, \beta \in \mathbb{C}[G]$ in an efficient manner. The method relies on a *Fast Fourier Transform* (FFT) of a group G, which breaks the group algebra into a collection of subalgebras whose elements span the entirety of $\mathbb{C}[G]$. The goal in this new framework

is to find small groups for which we can embed matrices in the group algebra, and the FFT gives a method for quickly multiplying them with a potential benefit in overall complexity.

However, when Cohn and Umans (2003) was published it was not yet known if the framework could be realized to even beat the naive $O(n^3)$ algorithm for matrix multiplication. Cohn and Umans sought the help of two others, Kleinberg and Szegedy, and eventually succeeded in using their ideas to duplicate the result of Coppersmith and Winograd (Cohn et al., 2005). The same paper also describes a rephrasing of a particular approach to the problem in purely combinatorial terms, using what they call *uniquely solvable puzzles*.

In this thesis, we provide a description of the framework of Cohn and Umans and find some new results that make more use of the computations in the group algebra. We then discuss the combinatorial problem involving uniquely solvable puzzles, providing a detailed overview and reinterpretation of the definitions along with a review of current progress in Cohn et al. (2005) and Alon et al. (2011). This is followed by a few new results in construction of uniquely solvable puzzles. At the end we provide a new open problem whose solution may lead to a matrix multiplication algorithm faster than Coppersmith-Winograd or the improvement in Williams (2011).

Chapter 2

Group-Theoretic Framework

This chapter introduces the group-theoretic approach of Cohn and Umans (2003) to matrix multiplication.

2.1 Group Algebras

Here we introduce the representation theory needed by Cohn and Umans (2003). The content follows the treatment of James and Liebeck (1993). A less didactic text with more focus on applying these ideas to algorithmic problems can be found in Bürgisser et al. (1997).

For the entirety of this section, G is a finite group. The group algebra $\mathbb{C}[G]$, where $\mathbb{C}$ denotes the complex numbers, is defined to be the set of formal linear combinations of elements of G with coefficients from the complex numbers, with componentwise addition and multiplication relying on the group operation in G. A general group algebra element takes the form $\alpha = \sum_{g \in G} \alpha_g g$, where $\alpha_g \in \mathbb{C}$. If we take a second arbitrary element $\beta = \sum_{g \in G} \beta_g g$, then the product and sum can be formally written as

$$\alpha + \beta = \sum_{g \in G} (\alpha_g + \beta_g) g,$$

$$\alpha \cdot \beta = \sum_{g \in G} \left(\sum_{h \in G} \alpha_h \beta_{h^{-1}g} \right) g.$$

Group algebras contain a surprising amount of structure. An important tool for studying them is to examine their $\mathbb{C}G$-submodules. A $\mathbb{C}G$-*module*

is a complex vector space V equipped with a definition for multiplication by group elements of G, so that vg is always defined for $v \in V, g \in G$. This multiplication is required to satisfy several axioms such as associativity and distributivity; the full definition is in section 4.2 of James and Liebeck (1993). The only $\mathbb{C}G$-modules we consider are the group algebra $\mathbb{C}[G]$ or its submodules. Notice that a $\mathbb{C}G$-module can also be treated as a vector space over $\mathbb{C}$, which gives us a natural way of defining its dimension. We call a nontrivial module V *irreducible* if its only submodules are the trivial one $\{0\}$ or V itself. Irreducible modules are important due to two results, Maschke's Theorem and Schur's Lemma, which are fundamental to representation theory. The first of these is given below; a proof can be found in Section 8.1 of James and Liebeck (1993).

Theorem 1 (Maschke's Theorem)**.** *Let G be a finite group, and let V be a $\mathbb{C}G$-module. If U is a $\mathbb{C}G$-submodule of V, then there exists another $\mathbb{C}G$-submodule W such that $V = U \oplus W$.*

This theorem allows us to decompose a larger $\mathbb{C}G$-module, such as the group algebra $\mathbb{C}[G]$ itself, by repeatedly breaking down the original module V into smaller and smaller submodules until we have eventually written it as a direct sum of irreducible modules. Maschke's Theorem guarantees we never get stuck in the process of doing this because every submodule U has a complement W.

While Maschke's Theorem guarantees we can do decompositions until we eventually are dealing with irreducible modules, our second result, Schur's Lemma, tells us why being able to consider irreducible modules gives us a lot of power. Before we state it, we define the notion of a $\mathbb{C}G$-*homomorphism*, which is simply a linear transformation from a $\mathbb{C}G$-module to another that also commutes with group elements. That is, if ϕ is a $\mathbb{C}G$-homomorphism from V to W, then $\phi(gv) = g\phi(v)$ for all $g \in G$ and $v \in V$. An *isomorphism* is as usual a bijective homomorphism. We will also give Schur's Lemma without proof; a reader interested in one may refer to Section 9.1 of James and Liebeck (1993).

Theorem 2 (Schur's Lemma)**.** *Let V and W be irreducible $\mathbb{C}G$-modules. If ϕ is a $\mathbb{C}G$-homomorphism, then either $\phi(v) = 0$ for all $v \in V$ or ϕ is a $\mathbb{C}G$-isomorphism. Furthermore, if $V = W$, then there exists $a \in \mathbb{C}$ such that $\phi(v) = av$ for all $v \in V$; that is, ϕ is a scalar multiple of the identity.*

Notice that if V and W are nonisomorphic $\mathbb{C}G$-modules, there are no nontrivial homomorphisms between them.

We give a quick example to show the power of Schur's Lemma. Let H be an abelian group, let V be an irreducible $\mathbb{C}G$-module, and choose an element $x \in H$. Consider the homomorphism from V to itself such that $\phi(v) = xv$; note that for this to be a homomorphism x must commute with all group elements, which is guaranteed to us by G being abelian. By Schur's Lemma, ϕ is an isomorphism and there exists a complex number λ_x such that $\phi(v) = vx = \lambda_x v$.

However, if we consider all vector subspaces of V, which need only be closed under scalar multiplication by complex numbers, then we see that every subspace is actually a submodule. By the assumption that V is irreducible, there are no nontrivial subspaces of V, so V as a vector space as dimension 1. Since V is arbitrary, any irreducible $\mathbb{C}H$-module of an abelian group H has dimension 1. This means that if we use Maschke's Theorem to decompose $\mathbb{C}[H]$ into irreducible modules as follows,

$$\mathbb{C}[H] = \bigoplus_{i=1}^{n} V_i,$$

where $V_1, V_2, \ldots, V_n$ are irreducible, then for an abelian group H, all of the V_i are one-dimensional as vector spaces. With more work, one can also show that V_i, V_j are isomorphic if and only if $i = j$.

The reason this decomposition into irreducible modules is important is because the irreducible modules can be used to construct a special basis of the group algebra $\mathbb{C}[G]$ for which most of the pairwise products of elements are zero. In the case of abelian groups this basis is simply an element of each irreducible module, and all the pairwise products are zero. By Schur's Lemma, two nonisomorphic irreducible modules V, W have no nontrivial homomorphisms, so the product of an element of V and an element of W is always 0. This means we only have to worry about multiplying elements of two irreducible modules which are isomorphic, and then combine the results at the end.

As an example of this process, let us examine the cyclic group $C_4 = \{1, g, g^2, g^3\}$. If we analyze the group algebra $\mathbb{C}[C_4]$, we can eventually decompose it into four irreducible modules defined as follows:

$$\begin{aligned}
V_1 &= \{a(1 + g + g^2 + g^3) \mid a \in \mathbb{C}\},\\
V_2 &= \{a(1 + ig - g^2 - ig^3) \mid a \in \mathbb{C}\},\\
V_3 &= \{a(1 - g + g^2 - g^3) \mid a \in \mathbb{C}\},\\
V_4 &= \{a(1 - ig - g^2 + ig^3) \mid a \in \mathbb{C}\}.
\end{aligned}$$

Here $i = \sqrt{-1}$. Therefore, for any group algebra element α, we can find complex numbers $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ such that

$$\begin{aligned}\alpha = \alpha_1(1 + g + g^2 + g^3) + \alpha_2(1 + ig - g^2 - ig^3) \\ + \alpha_3(1 - g + g^2 - g^3) + \alpha_4(1 - ig - g^2 + ig^3).\end{aligned}$$

If we similarly represent group algebra element β using complex numbers $\beta_1, \beta_2, \beta_3, \beta_4$, then we obtain that

$$\begin{aligned}\alpha\beta = 4\alpha_1\beta_1(1 + g + g^2 + g^3) + 4\alpha_2\beta_2(1 + ig - g^2 - ig^3) \\ + 4\alpha_3\beta_3(1 - g + g^2 - g^3) + 4\alpha_4\beta_4(1 - ig - g^2 + ig^3).\end{aligned}$$

The multiplication of two arbitrary elements of $\mathbb{C}[C_4]$, which is $O(n^2)$ when done in the naive fashion, has been simplified into the multiplication of four complex numbers and is now $O(n)$. This incredible speedup of group algebra multiplication that the irreducible submodules give us is the central idea of the approach to fast matrix multiplication by Cohn and Umans (2003).

For nonabelian groups, some complications are introduced. Irreducible modules may have dimensions higher than 1, and indeed for any positive integer n there exists a group G whose group algebra has an irreducible module of dimension at least n. In the case of an irreducible submodule V of $\mathbb{C}[G]$ such that V has dimension n, we find that in the decomposition of $\mathbb{C}[G]$ into irreducible submodules there are always exactly n of these submodules isomorphic to V, which we call $V_1, V_2, \ldots, V_n$. This makes the multiplication of two group algebra elements a little more complicated than simply $|G|$ complex number products, but one can show that the appropriate arithmetic for these n submodules can be expressed as the product of two $n \times n$ matrices. This applies to the case $n = 1$ also, which we had above, because we can view the multiplication of complex numbers as simply the product of two 1×1 complex matrices.

All of this culminates in the following central result of the representation theory of finite groups. This result appears as 13.36 of Bürgisser et al. (1997), which does not provide a proof but notes that one can be found in Serre (1986).

Theorem 3 (Wedderburn Decomposition Theorem)**.** *For a finite group G, $\mathbb{C}[G]$ can be decomposed into matrix rings,*

$$\mathbb{C}[G] \cong \bigoplus_i \mathbb{C}^{d_i \times d_i},$$

where the d_i are the degrees of the distinct irreducible representations of $\mathbb{C}[G]$.

The isomorphism of the Wedderburn Theorem is referred to as a *Discrete Fourier Transform* or DFT. An algorithm for quickly doing the decomposition is referred to as a *Fast Fourier Transform* or FFT.

From the standpoint of algorithmic complexity, knowing the highest degree of an irreducible submodule is important to determining how quick the multiplication can be done. Except for the fact that abelian groups have all $d_i = 1$, getting a handle on the degrees of an arbitrary group G is hard. However, since taking the dimensions of both sides of the above equality gives us

$$|G| = \sum_{i=1}^{n} d_i^2,$$

we can be sure that the degree of every irreducible representation is at most $\sqrt{|G|}$.

The cornerstone of the ideas for matrix multiplication algorithms that we are focusing on makes use of the FFT to do the difficult work for us. This is the idea introduced in Cohn and Umans (2003). The linear algebra operations needed to perform and work with the embedding of matrices into $\mathbb{C}[G]$ do not come for free. As $\mathbb{C}[G]$ is a vector space of dimension $|G|$, in general the smaller we can get the group G the more the complexity improves. So in order to come up with better matrix multiplication algorithms, all one has to do is find a group G of small size and an embedded structure of matrix multiplication.

As an example of the use of the FFT, in Cohn and Umans (2003) a method for fast polynomial multiplication is described which works by embedding the problem in a group algebra in which polynomial multiplication can be naturally expressed. If the degree of the polynomials are n, then the group G can be chosen to be the cyclic group C_k for $k \geq 2n + 1$. The reason this works is that multiplying two polynomials is done by computing cyclic convolutions of coefficients. In particular, suppose $P(x) = \sum_{i=0}^{n} a_i x^i$, $Q(x) = \sum_{i=0}^{n} b_i x^i$, and $R(x) = P(x)Q(x) = \sum_{i=0}^{2n} c_i x^i$. Then

$$c_k = \begin{cases} \sum_{i=0}^{k} a_i b_{k-i} & k \leq n, \\ \sum_{i=k-n}^{n} a_i b_{k-i} & k > n. \end{cases}$$

In the group algebra $\mathbb{C}[C_k]$, multiplying two elements has this same structure to it; we can simply treat the x in $P(x), Q(x), R(x)$ as the generator of C_k, which gives an exact correspondence between multiplication of polynomials and multiplication in the group algebra $\mathbb{C}[G]$. Because we picked $k \geq 2n + 1$, all of the monomials in $R(x)$ are necessarily distinct in $\mathbb{C}[G]$.

Now that we have the embedding, we use the FFT of $\mathbb{C}[G]$ to compute the product PQ within $\mathbb{C}[G]$ in a fast manner. The resulting algorithm has complexity $O(n \log n)$, a huge improvement over the naive algorithm which requires $O(n^2)$ operations.

The original problem of multiplying polynomials can also be expressed as a problem of multiplying special matrices. If $P(x) = \sum_{i=0}^{n} a_i x^i$ and $Q(x) = \sum_{i=0}^{n} b_i x^i$ are degree n polynomials, then we can rewrite these as particular $k \times k$ circulant matrices with $k \geq 2n + 1$:

$$A = M(P) = \begin{pmatrix} a_0 & a_1 & \cdots & a_n & 0 & \cdots & 0 \\ 0 & a_0 & \cdots & a_{n-1} & a_n & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_0 & a_1 & \cdots & a_n \\ 0 & 0 & \cdots & 0 & a_0 & \cdots & a_{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & \cdots & 0 & 0 & \cdots & a_0 \end{pmatrix},$$

$$B = M(Q) = \begin{pmatrix} b_0 & b_1 & \cdots & b_n & 0 & \cdots & 0 \\ 0 & b_0 & \cdots & b_{n-1} & b_n & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & b_0 & b_1 & \cdots & b_n \\ 0 & 0 & \cdots & 0 & b_0 & \cdots & b_{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ b_1 & b_2 & \cdots & 0 & 0 & \cdots & b_0 \end{pmatrix}.$$

(Here the lower left corners of each matrix contain a triangle of potentially nonzero coefficients with bottom row reading $a_1, a_2, \ldots, a_n$ and each successive row omitting one element from the beginning.)

The product AB gives a new $k \times k$ circulant matrix, and the coefficients of the polynomial obtained by multiplying $P(x)Q(x)$ can be read off of the first row of this resulting matrix, in the same way that the coefficients of $P(x)$ can be read from the first row of A. Note that this does not create a full embedding of polynomials into a set of matrices, since the dimensions of each matrix depend on the degree of the polynomials being multiplied. But the goal is to obtain an algorithm that can efficiently handle any individual instance of polynomial multiplication, and the above scheme is powerful enough for this purpose.

That this particular form of matrix has a highly efficient multiplication is a special case of a more general result. Let $G = \{g_1, g_2, \ldots, g_r\}$ be a finite

group of r elements. We can define the matrix

$$A = \begin{pmatrix} a_{g_1 g_1^{-1}} & a_{g_1 g_2^{-1}} & a_{g_1 g_3^{-1}} & \cdots & a_{g_1 g_r^{-1}} \\ a_{g_2 g_1^{-1}} & a_{g_2 g_2^{-1}} & a_{g_3 g_3^{-1}} & \cdots & a_{g_2 g_r^{-1}} \\ a_{g_3 g_1^{-1}} & a_{g_3 g_2^{-1}} & a_{g_3 g_3^{-1}} & \cdots & a_{g_3 g_r^{-1}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{g_r g_1^{-1}} & a_{g_r g_2^{-1}} & a_{g_r g_3^{-1}} & \cdots & a_{g_r g_r^{-1}} \end{pmatrix}.$$

where of course $a_g = a_h$ if $g = h$ in the group G. This is known as a *G-circulant* matrix (Diaconis, 1988). A way to characterize it is that if we write out a group multiplication table for G as below, with inverses in the columns, the result is structured identically to A.

	g_1^{-1}	g_2^{-1}	g_3^{-1}	$\cdots$	g_r^{-1}
g_1	$g_1 g_1^{-1}$	$g_1 g_2^{-1}$	$g_1 g_3^{-1}$	$\cdots$	$g_1 g_r^{-1}$
g_2	$g_2 g_1^{-1}$	$g_2 g_2^{-1}$	$g_3 g_3^{-1}$	$\cdots$	$g_2 g_r^{-1}$
g_3	$g_3 g_1^{-1}$	$g_3 g_2^{-1}$	$g_3 g_3^{-1}$	$\cdots$	$g_3 g_r^{-1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
g_r	$g_r g_1^{-1}$	$g_r g_2^{-1}$	$g_r g_3^{-1}$	$\cdots$	$g_r g_r^{-1}$

Furthermore, there exists a straightforward way to embed A into the group algebra $\mathbb{C}[G]$; if we let f be this embedding, then

$$f(A) = a_{g_1} \cdot g_1 + a_{g_2} \cdot g_2 + \cdots + a_{g_r} \cdot g_r,$$

and the essential property that $f(A)f(B) = f(AB)$ holds. In fact, the naive methods for both multiplying two group algebra elements and multiplying two matrices will obtain the same linear combinations of elements in the results and do the same additions and multiplications. But since we have the FFT for multiplying elements in $\mathbb{C}[G]$, the embedding leads to an improvement in complexity. Notice that the example of polynomial multiplication is a specific case of this for when the group is cyclic.

One question that may arise at this point is the cost of these operations with regard to practical use of matrix multiplication algorithms. In Gonzalez-Sanches et al. (2009) the observation is made that the computations associated with transforming the matrices into group algebra elements and extracting the product matrix out of the result are extremely expensive in practice, although from the standpoint of theoretical computer science they are merely constants to be ignored. Some of the problems with

practical use may be ameliorated by taking advantage of the result that group-based matrix multiplication algorithms are stable, proven in Holtz et al. (2007), which makes numerical approximations viable, and the authors of Gonzalez-Sanches et al. (2009) present some approaches to improving some of the intermediate computations. But currently practical implementations of these group-based algorithms are infeasible. Coppersmith and Winograd's algorithm is also inviable in practice, and Strassen's algorithm continues to be the most efficient algorithm for multiplying large matrices in actual applications (Coppersmith and Winograd, 1990; Robinson, 2005). The matrix multiplication algorithms with the best known complexity are currently only of theoretical importance.

2.2 Triple Product Property

This section introduces the methods used to allow the FFT to be used on the problem of multiplying two arbitrary square matrices. This section follows Cohn and Umans (2003), the original paper where the ideas were introduced.

For the problem of multiplying arbitrary $n \times n$ matrices, the group-theoretic approach proceeds by embedding the matrices into a group algebra $\mathbb{C}[G]$ which has a multiplicative structure matching that of matrix multiplication and using the group's FFT to obtain a fast algorithm. Before giving the full general method, we begin with an example of a group that contains the needed structure for multiplying 2×2 matrices.

Consider the group D_8, the dihedral group with eight elements. One way to view the elements of this group is as the symmetries of a square, with both rotations and reflections allowed. Another way is to use the group's presentation, which is that there are two generators x and y satisfying $x^4 = 1, y^2 = 1, xy = yx^{-1}$. All elements of the group can be expressed as a product of a sequence of x, x^{-1}, y, y^{-1} factors, with two elements equal if they can be shown equal through the aforementioned identities. These two perspectives on the group are related by the fact that x can be viewed as a 90 degree rotation and y as a reflection. Indeed, one can check geometrically that if a 90 degree rotation is performed and then followed by a reflection, then this is the same as doing the reflection first and then the 90 degree rotation in the other direction, which is a description of the identity $xy = yx^{-1}$.

We want to use D_8 in order to multiply two 2×2 matrices. Let these

two matrices be A and B, with the following entries and product:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \qquad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix},$$
$$AB = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}.$$

Consider the following 2-element subsets of D_8: $S = \{1, x\}$, $T = \{1, y\}$, $U = \{1, x^2y\}$. We embed A and B into the group algebra $\mathbb{C}[D_8]$ as follows:

$$A = \begin{array}{c} \\ 1 \\ x^{-1} \end{array} \begin{array}{c} \begin{array}{cc} 1 & y \end{array} \\ \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \end{array} \mapsto \alpha = a_{11} \cdot 1 + a_{12}y + a_{21}x^{-1} + a_{22}x^{-1}y,$$
$$B = \begin{array}{c} \\ 1 \\ y^{-1} \end{array} \begin{array}{c} \begin{array}{cc} 1 & x^2y \end{array} \\ \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \end{array} \mapsto \beta = b_{11} \cdot 1 + b_{12}x^2y + b_{21}y + b_{22}x^2.$$

Here we obtain α by making a_{ij} the coefficient of the inverse of the ith element of S times the jth element of T. Likewise we obtain β by putting b_{ij} in front of the inverse of the ith element of T times the jth element of U. When we compute $\alpha\beta$, this construction results in all the terms of each entry in our matrix product being collected together in front of the same group element, as shown below:

$$\begin{aligned} \alpha\beta &= (a_{11} \cdot 1 + a_{12}y + a_{21}x^{-1} + a_{22}x^{-1}y)(b_{11} \cdot 1 + b_{12}x^2y + b_{21}y + b_{22}x^2) \\ &= (a_{11}b_{11} + a_{12}b_{21}) \cdot 1 + (a_{11}b_{12} + a_{12}b_{22})x^2y + (a_{21}b_{11} + a_{22}b_{21})x^{-1} \\ &\quad + (a_{21}b_{12} + a_{22}b_{22})xy + (a_{11}b_{21} + a_{12}b_{11}) \cdot y + (a_{11}b_{22} + a_{12}b_{12})x^2 \\ &\quad + (a_{21}b_{21} + a_{22}b_{11})x^{-1}y + (a_{21}b_{22} + a_{22}b_{12})x. \end{aligned}$$

Notice that the first four coefficients here are the entries of our product matrix AB after combining all like terms. As a result, we can compute AB simply by reading off the coefficients of these group elements, all of which take the form of an inverse of an element of S times an element of U. This is no accident: if we generalize the above construction and let $S = \{s_1, s_2, \ldots, s_m\}$, $T = \{t_1, t_2, \ldots, t_n\}$, $U = \{u_1, u_2, \ldots, u_p\}$, which means A has dimensions $m \times n$ and B has dimensions $n \times p$, then in $\alpha\beta$ the term $a_{ij}b_{jk}$ is in front of the group element $s_i^{-1}t_jt_j^{-1}u_k = s_i^{-1}u_k$, and indeed $a_{ij}b_{jk}$ is one of the terms in the ith row and kth column of the product matrix. The

construction very deliberately forces the correct sum of terms to show up in the product of group algebra elements.

However, there is an issue that can crop up. While in the above example the only terms to show up in the coefficients of the group elements we read the product from were the ones we needed, a bad choice of S, T, U can lead to extra terms showing up. If instead of $U = \{1, x^2y\}$ we had chosen $U' = \{1, xy\}$ and used it and T to embed the matrix B as β', then

$$\begin{aligned}\alpha\beta' &= (a_{11} \cdot 1 + a_{12}y + a_{21}x^{-1} + a_{22}x^{-1}y)(b_{11} \cdot 1 + b_{12}xy + b_{21}y + b_{22}x^{-1}) \\ &= (a_{11}b_{11} + a_{12}b_{21}) \cdot 1 + (a_{11}b_{12} + a_{12}b_{22})xy \\ &\quad + (a_{21}b_{11} + a_{22}b_{21} + a_{11}b_{22} + a_{12}b_{12})x^{-1} \\ &\quad + (a_{21}b_{12} + a_{22}b_{22} + a_{11}b_{21} + a_{12}b_{11})y \\ &\quad + (a_{21}b_{21} + a_{22}b_{11})x^{-1}y + (a_{21}b_{22} + a_{22}b_{12})x^2.\end{aligned}$$

Here some of the entries in our product, specifically $a_{21}b_{11} + a_{22}b_{21}$ and $a_{21}b_{12} + a_{22}b_{22}$, have been lumped together with other terms that do not appear in the matrix product at all, and obtaining the entries of the product matrix is no longer possible to do efficiently.

This leads naturally to the question of what went wrong here and how can it be avoided. As before, let $S = \{s_1, s_2, \ldots, s_m\}$, $T = \{t_1, t_2, \ldots, t_n\}$, $U = \{u_1, u_2, \ldots, u_p\}$. If we could show that any 6-tuple of indices i, j, k, i', j', k' with $s_i^{-1}t_j t_{j'}^{-1}u_k = s_{i'}^{-1}u_{k'}$ implies that $i = i', j = j', k = k'$, then this would be enough to show that the only terms in the coefficient of $s_i^{-1}u_k$ are those that we are looking for. As it turns out, the triple $S = \{1, x\}$, $T = \{1, y\}$, $U = \{1, x^2y\}$ does satisfy this property. However, the triple $S = \{1, x\}$, $T = \{1, y\}$, $U' = \{1, xy\}$ does not satisfy this property since we have $1 \cdot y \cdot 1 \cdot 1 = x^{-1} \cdot xy$, among other counterexamples.

The condition above is called the Triple Product Property in Cohn and Umans (2003). We now give its formal and general definition.

Definition 1. *Let G be a finite group. For S a nonempty subset of G, define $Q(S)$, the quotient of S, to be all elements of the form $s_1 s_2^{-1}$ where $s_1, s_2 \in S$. We say that the triple of subsets $S_1, S_2, S_3 \subseteq G$ satisfy the* triple product property *if $q_1q_2q_3 = 1$ implies $q_1 = q_2 = q_3 = 1$, where $q_i \in Q(S_i)$ for $i = 1, 2, 3$.*

As an example of working with this definition, we give here a full technical demonstration of why our earlier triple of subsets for $G = D_8$, with $S_1 = \{1, x\}$, $S_2 = \{1, y\}$, and $S_3 = \{1, x^2y\}$, satisfies this property. Note that S_2 and S_3 are subgroups, so $Q(S_2) = S_2$ and $Q(S_3) = S_3$. Furthermore their intersection is trivial, so if we consider the solutions to $q_2q_3 = 1$ with

$q_i \in Q(S_i)$, taking $q_2 \neq 1$ would force the product to be contained in the coset $q_2 S_3$, which is not equal to S_3 because $q_2 \notin S_3$. The same holds for $q_3 \neq 1$.

Finally, let H be the subgroup generated by S_2, S_3, and observe that $Q(S_1) = \{1, x, x^3\}$. Again, we have $H \cap Q(S_1)$ trivial, so if we choose $q_1 \neq 1$ in $q_1 q_2 q_3 = 1$, the product is contained in $q_1 H$ and cannot be the identity. So $q_1 = 1$, and combining this with the work of the previous paragraph finishes the proof. Therefore $q_1 q_2 q_3 = 1$ implies $q_1 = q_2 = q_3 = 1$ and the triple product property is satisfied.

Notice that there is no requirement that any of the subsets be a subgroup, as in the case of S_1 here, although as we have seen it can simplify the arguments needed to prove the property holds for a given triple.

As our work in the previous example suggests, finding a triple of subsets satisfying this property immediately leads to a way of embedding matrices into the group algebra so that we may use the FFT. The next result, which is Theorem 2.5 in Cohn and Umans (2003), makes this precise.

Theorem 4. *Suppose G is a group with subsets S_1, S_2, S_3 satisfying the triple product property. Then the complexity of multiplying $|S_1| \times |S_2|$ and $|S_2| \times |S_3|$ matrices is no worse than the complexity of multiplying two elements of the group algebra $\mathbb{C}[G]$.*

Therefore, to obtain a fast matrix multiplication algorithm for $n \times n$ matrices, we need only exhibit a small group G and three n-element subsets satisfying the triple product property.

2.3 Extraneous Group Algebra Computations

This section presents new results, particularly Theorem 5, about making additional use of the triple product property and the group algebra computations.

Let us return to our example using the group $G = D_8$, with subsets $S_1 = \{1, x\}$, $S_2 = \{1, y\}$, $S_3 = \{1, x^2 y\}$:

$$\begin{aligned}
\alpha\beta &= (a_{11} \cdot 1 + a_{12} y + a_{21} x^{-1} + a_{22} x^{-1} y)(b_{11} \cdot 1 + b_{12} x^2 y + b_{21} y + b_{22} x^2) \\
&= (a_{11} b_{11} + a_{12} b_{21}) \cdot 1 + (a_{11} b_{12} + a_{12} b_{22}) x^2 y + (a_{21} b_{11} + a_{22} b_{21}) x^{-1} \\
&\quad + (a_{21} b_{12} + a_{22} b_{22}) xy + (a_{11} b_{21} + a_{12} b_{11}) \cdot y + (a_{11} b_{22} + a_{12} b_{12}) x^2 \\
&\quad + (a_{21} b_{21} + a_{22} b_{11}) x^{-1} y + (a_{21} b_{22} + a_{22} b_{12}) x.
\end{aligned}$$

When multiplying $\alpha\beta$ in the group algebra, we have done a number of other computations that are remaining completely unused. This is unsurprising since when taking the product of two matrices in general we do not require the product of every possible pair of entries. So while the coefficients on the other four group elements in $\alpha\beta$ are necessarily computed when we do the multiplication within $\mathbb{C}[D_8]$, they are unused by our algorithm. In general, because each $n \times n$ matrix results in n^2 terms when embedded into the group algebra, their product in the group algebra has n^4 scalar products, and we only need n^3 of them to compute the product matrix. So we are computing n times as many products as we need.

However, for this particular example the coefficients actually have some structure to them also. In particular, if we define

$$A' = \begin{matrix} \\ 1 \\ x^{-1} \end{matrix} \begin{matrix} \begin{matrix} y & 1 \end{matrix} \\ \begin{pmatrix} a_{12} & a_{11} \\ a_{22} & a_{21} \end{pmatrix} \end{matrix},$$

where the column labels have been multiplied by y, then we can notice that the unused coefficients of $\alpha\beta$ show up perfectly in

$$A'B = \begin{matrix} \\ y \\ x^3y \end{matrix} \begin{matrix} \begin{matrix} 1 & x^2y \end{matrix} \\ \begin{pmatrix} a_{11}b_{21} + a_{12}b_{11} & a_{11}b_{22} + a_{12}b_{12} \\ a_{21}b_{21} + a_{22}b_{11} & a_{21}b_{22} + a_{22}b_{12} \end{pmatrix} \end{matrix}.$$

Here the only change we have made to the group element indexing is to multiply both row labels by y. This is a result of a certain amount of structure present in the triple product subsets we constructed.

The following theorem shows what conditions need to be satisfied in order for us to find such structure in the unused terms of the group algebra multiplication. The conditions involve the normalizer N_G: recall that $N_G(S)$ for a subset $S \subseteq G$ is the set of all elements g such that $gsg^{-1} \in S$ for all $s \in S$.

Theorem 5. *Suppose $S_1, S_2, S_3 \subseteq G$ satisfy the triple product property allowing us to multiply $|S_1| \times |S_2|$ and $|S_2| \times |S_3|$ matrices, with the further conditions that S_2 is a subgroup and that either S_1 or S_3 is a subset of $N_G(S_2)$. Let A and B be matrices of these dimensions, respectively, and let α andβ be their group algebra embeddings in $\mathbb{C}[G]$. Let $m = |S_1| \cdot |S_3|$, the number of entries in AB. Then there exists a set T of $|S_2|$ disjoint m-term sequences of group elements of G, such that for each sequence $g_1, g_2, \ldots, g_m$ in T, the coefficients of $g_1, g_2, \ldots, g_m$ in the group algebra element $\alpha\beta$ give the entries in the product of a permutation of the*

columns of A with B. The sequence of T that contains the identity is the one that corresponds to the actual product AB.

Proof. Let $S_2 = \{t_1, t_2, \ldots, t_n\}$ (so $|S_2| = n$) in such a way that the columns of A are labeled $t_1, t_2, \ldots, t_n$ in that order. Choose an arbitrary t_i. Permute the columns of A to get the matrix A' so that the column labels become $t_i^{-1}t_1, t_i^{-1}t_2, \ldots, t_i^{-1}t_n$, but leave B unchanged. We claim $\alpha\beta$ contains the entries of $A'B$ as coefficients. Specifically, take the m group elements $s^{-1}t_i^{-1}u$ for $s \in S_1, u \in S_3$. Verifying the right terms show up in front of these group elements in $\alpha\beta$ is a straightforward consequence of the construction.

What remains to be shown is that if $s^{-1}t_iu = s'^{-1}t_ju'$, then $s = s', t_i = t_j, u = u'$, meaning that there is no interference in the construction above. Without loss of generality, take $S_1 \subseteq N_G(S_2)$; otherwise we may just invert the whole product. Rewrite this as $t_j^{-1}s's^{-1}t_iuu'^{-1} = e$. Since $S_1 \subseteq N_G(S_2)$, $t_j^{-1}s's^{-1} = s's^{-1}t_j'$ for some $t_j' \in S_2$. So our equation becomes $s's^{-1}t_j't_iuu'^{-1} = 1$. Since $t_j't_i \in S_2 = Q(S_2)$, by the Triple Product Property the only solution to this is if $s's^{-1} = 1, t_j't_i = 1, uu'^{-1} = 1$. So in particular $s = s', u = u'$. It is straightforward to see that $t_i = t_j$ as a consequence of this and the original equality. □

Although the condition regarding the normalizer $N_G(S_2)$ is awkward and significantly reduces the elegance of the result, it is a necessary hypothesis. We present a counterexample that only fails the normalizer condition and does not satisfy the conclusions of the theorem. Consider the group $G = D_6$, the dihedral group of 6 elements with generators x, y satisfying $x^3 = 1, y^2 = 1, xy = yx^{-1}$. This group can alternately be viewed as S_3, the permutation group of a three element set. Take the three subsets:

$$S = \{1, xy\}, \qquad T = \{1, y\}, \qquad U = \{1, x^2y\}.$$

We first verify the triple product property. All three of these subsets are subgroups, so $Q(S) = S, Q(T) = T, Q(U) = U$. So for these subsets to satisfy the triple product property, we need $stu = 1$ for $s \in S, t \in T, u \in U$ to imply that $s = t = u = 1$. One can check easily that of the eight possible ways to choose s, t, u, only $s = t = u = 1$ gives a product of 1. On the other hand, observe that $xy \cdot y \cdot x^2y = 1 \cdot y \cdot 1$, which along the lines of the proof of Theorem 5 gives a solution to $s^{-1}t_iu = s'^{-1}t_ju'$ that has $s \neq s'$ and $u \neq u'$. So despite T being a subgroup and S, T, U satisfying the triple product property, we are not able to use the theorem's method of making use of all n^4 terms of the group algebra product.

Here is another such counterexample, which makes use of the group $G = S_4$, the permutation group of a four element set. The elements are written in terms of cycle notation:

$$S = \{1, (14)\}, T = \{1, (123), (132)\}, U = \{1, (34)\}.$$

We leave the details of verifying that this is another counterexample to the reader.

The theorem in the previous section can also be phrased in a more general and elegant fashion if we work with tensors, which can be viewed as a multidimensional generalization of vectors and matrices. The setup here follows Kolda and Bader (2009).

Definition 2. *A* tensor *of order k is a k-dimensional array of elements of a field F. Each dimension is referred to as a* mode. *Let $\mathcal{X}$ be such a tensor, and suppose its dimensions are $d_1, d_2, \ldots, d_k$. We write $\mathcal{X} \in F^{d_1 \times d_2 \times \cdots \times d_k}$. We write each entry of $\mathcal{X}$ as $x_{i_1, i_2, \cdots, i_k}$ for $1 \leq i_j \leq d_j$ for each $j = 1, 2, \ldots, k$.*

A fiber *of a tensor $\mathcal{X}$ is a one-dimensional subarray of $\mathcal{X}$ and can be viewed as the analogue of rows and columns of matrices. Formally, they can be written as the elements $x_{i_1, i_2, \cdots, i_k}$ where all of the indices i_j are fixed except for one of them, and this particular i_j is allowed to run from 1 to d_j. This would be one of the* mode j fibers.

A slice *of a tensor is a two-dimensional subarray of $\mathcal{X}$, which on its own forms a matrix. Similar to the above, they can be expressed formally as the elements $x_{i_1, i_2, \cdots, i_k}$ where all but two of the indices are fixed.*

Due to their complicated structure, defining multiplication of two tensors which are of order three or higher is much more awkward than definitions of ordinary matrix multiplication and is outside the scope of this paper. However, the multiplication of tensors by matrices or vectors is within reach and useful for our purposes.

Definition 3. *Let $\mathcal{X}$ be a tensor of order k with dimensions $d_1, d_2, \ldots, d_k$. Let j be a positive integer less than or equal to k. The* j-mode matrix product *of the tensor $\mathcal{X}$ with a matrix M in $F^{d_j \times d'}$ is defined by taking each of the mode j fibers of $\mathcal{X}$ as vectors and replacing each one by its product with M in the tensor. The resulting tensor, written as $\mathcal{X} \times_j M$, is an order k tensor contained in $F^{d_1 \times d_2 \times \cdots \times d_{j-1} \times d' \times d_{j+1} \times \cdots \times d_k}$.*

This language allows us to provide a restatement of Theorem 5.

Theorem 6. *Let G be a finite group and let S_1, S_2, S_3 be subsets of G satisfying the triple product property where S_2 is a subgroup of G and either S_1 or S_3 is a subset of $N_G(S_2)$. Suppose that M is an $|S_2| \times |S_3|$ matrix, $\mathcal{X}$ is a tensor of order 3 with dimensions $|S_1|, |S_2|, |S_2|$ (so $\mathcal{X} \times_3 M$ has dimensions $|S_1|, |S_2|, |S_3|$), and the slices along modes 2 and 3 all form S_2-circulant matrices. Then using the group algebra and triple product property to multiply M by any mode 1,2 slice of $\mathcal{X}$ computes the entire 3-mode matrix product $\mathcal{X} \times_3 M$.*

Note that in constructing the tensor $\mathcal{X}$, we stack A and its permuted forms along modes 1 and 2. It can be checked that the constraints on the permutations of A in the statement of Theorem 5 are equivalent to the mode 2 and 3 slices of $\mathcal{X}$ being S_2-circulant. Therefore, this is equivalent to Theorem 5, with the benefit of being a bit more obvious about what the $|S_2|$ disjoint sequences are. Also note that in computing this tensor we make use of all $|S_1||S_2|^2|S_3|$ terms that were calculated in the product of the two group algebra elements, so this result is the best possible in terms of how much more computation we can get for free.

A natural question to ask at this point is whether Theorem 5 has a natural application. In situations where fast computation of matrices is desired, is there any situation in which we might also be interested in also multiplying by the permuted matrices that appear in the statement of the theorem? At the moment this remains an open question.

Another possible approach to making use of Theorem 5 has to do with finding a special class of matrices for which there are algorithms based on the triple product property that can multiply these matrices particularly quickly. Assuming that such methods existed so that we can quickly multiply an arbitrary matrix A by a certain class of matrices from a set X in a way that allows us to use Theorem 5, we might be able to obtain speedups for a much broader class of matrices. If for a matrix $B \in X$ we let $B = B_1$ and take $B_1, B_2, B_3, \ldots, B_n$ to be the permuted forms, then for any linear combination of these matrices,

$$C = c_1 B_1 + c_2 B_2 + \cdots + c_n B_n,$$

we have a very fast algorithm for quickly multiplying AC. After obtaining the decomposition above, we multiply A by B_1, which we assumed to be fast. This obtains for free the products $AB_1, AB_2, \ldots, AB_n$. We then sum up these products with the appropriate coefficients to obtain the final product AC.

The only question is what such a class of matrices X might look like. One natural guess is those matrices which have a zero coefficient in front

of the highest degree irreducible submodules in the FFT. In investigating this, two computations were done using the symmetric group S_3 and S_4. For S_3, the triple product subsets were the following two-element subsets:

$$S_1 = \{1, (12)\}, \quad S_2 = \{1, (13)\}, \quad S_3 = \{1, (23)\}.$$

The degrees of the irreducible modules of S_3 are $1, 1$, and 2. When one takes an arbitrary 2×2 matrix, uses the above subsets to embed it into $\mathbb{C}[S_3]$, and projects it onto the subspace orthogonal to the two-dimensional representation, the zero matrix is always obtained. For S_4, the results were similar. The degrees are $1, 1, 2, 3$, and 3, so we attempted to project onto the subspace orthogonal to both three-dimensional representations. Using the following three-element subsets gave that only the zero matrix embedded into this subspace:

$$S_1 = \{1, (123), (132)\}, \quad S_2 = \{1, (124), (142)\}, \quad S_3 = \{1, (1324), (1423)\}.$$

So these computations suggest this approach is much too restrictive. However, in these examples we eliminated a particularly large portion of the group algebra, reducing the dimension by at least a factor of three. If the size of our matrices is $n \times n$, then the space V formed by the image of embedding all possible 3×3 matrices forms a subspace of $\mathbb{C}[G]$ with dimension n^2. If we eliminate irreducible modules with dimensions $d_1, d_2, \ldots, d_k$ then the orthogonal subspace W we are projecting onto has dimension $|G| - d_1^2 - d_2^2 - \cdots - d_k^2$. It is a simple result of linear algebra that $V \cap W$ has dimension at least $\dim V + \dim W - |G|$ because they are both subspaces of $\mathbb{C}[G]$ which has dimension $|G|$. Therefore, if we choose $d_1, d_2, \ldots, d_k$ so that $d_1^2 + d_2^2 + \cdots + d_k^2 < n^2$, we can guarantee that $V \cap W$ is a nontrivial vector space of matrices for which we have a faster algorithm.

Another possible refinement of the above example is to have the components of the highest degree irreducible modules become diagonal matrices instead of the zero matrix. Besides increasing the dimension of the orthogonal subspace we are projecting onto without a big hit to performance, the results of this approach would change with the choice of the isomorphic copies of the irreducible submodules we are attempting to get rid of, and we could tailor this choice to maximize the dimension of the set X of matrices that we obtain in the final computation.

Chapter 3

Uniquely Solvable Puzzles

Two years after Cohn and Umans (2003), a follow-up paper Cohn et al. (2005) with the same authors along with Balasz and Szegedy was published. This paper introduced a combinatorial approach, that of constructing uniquely solvable puzzles (USPs), to finding matrix multiplication algorithms. In this chapter we provide an exposition and some new results regarding USPs.

3.1 Definition and Understanding

USPs are one approach available to attacking the problem of constructing groups and subsets satisfying the triple product property. The motivation of USPs is that one family of groups that has given promising results are semidirect products of abelian groups with symmetric groups. USPs correspond to a particular case of this construction in a way that allows for phrasing all the needed constraints in a combinatorial setting.

The formal definition for USPs that can be found in Cohn et al. (2005) is particularly opaque and difficult to get a handle on. Below we give this definition as it appeared originally, and then we spend some time discussing it in more concrete terms and illuminating the intuitions one should have about USPs.

Definition 4. *A* uniquely solvable puzzle (USP) *of width k is a subset U of $\{1,2,3\}^k$ satisfying the following property. Let $Sym(U)$ denote the ways of permuting the elements of U, so that it is isomorphic to $S_{|U|}$. The condition is that for all permutations $\pi_1, \pi_2, \pi_3 \in Sym(U)$, either we have $\pi_1 = \pi_2 = \pi_3$ or there exists a $u \in U$ and $i \in \{1,2,\ldots,k\}$ such that for at least two $j \in \{1,2,3\}$, $(\pi_j(u))_i = j$ is true.*

3	3	3	3	3	3
1	3	3	2	3	3
3	1	3	3	2	3
1	1	3	2	2	3
3	3	1	3	3	2
1	3	1	2	3	2
3	1	1	3	2	2
1	1	1	2	2	2

Figure 3.1 Example USP from Cohn et al. (2005).

The next few pages explain and reinterpret this definition in much more intuitive terms. First, we provide some auxiliary definitions. These do not introduce any new concepts and merely provide another perspective on the content of Definition 4.

Definition 5. *A* USP candidate *of width k is a subset U of $\{1,2,3\}^k$, which we imagine as a grid with k columns and $|U|$ unordered rows each of whose entries is from the set $\{1,2,3\}$. For $u \in U, i \in \{1,2,\ldots,k\}$, we refer to the element in the ith column of the row u by writing u_i.*

Let U be a USP candidate and let $(\pi_1, \pi_2\pi_3)$ be three permutations of the rows of U. Let u_i be an element of the grid U. We say (π_1, π_2, π_3) satisfy the USP rule *for this position if for at most one $j \in \{1,2,3\}$ we have $(\pi_j(u))_i = j$ is true.*

It is straightforward to prove from these definitions that a USP candidate U is then a USP if and only if a triple of permutations (π_1, π_2, π_3) satisfying the USP rule for all positions implies that $\pi_1 = \pi_2 = \pi_3$. In this way we can think of a USP as being a puzzle which has only one solution under the constraints of the USP rule, and a USP candidate as an object of the same structure but which may or may not have a unique solution.

Below is an example USP viewed with the grid visualization described in Definition 5. This example is the same as the one presented in Cohn et al. (2005). It is a USP of width 6 and size 8.

We now turn our attention to the USP rule. The first thing to note is that the constraints it imposes on π_j for $j \in \{1,2,3\}$ mean that for π_j, only the js in the grid are important. That is, it only matters whether the number contained in $(\pi_j(u))_i$ is a j or not. Also, while the USP rule speaks in terms of the permutations π_1, π_2, π_3 moving around entire rows of the table, it is better to think of them as moving around the fragments of rows that contain only their own number. That is, when π_j is permuting rows, we

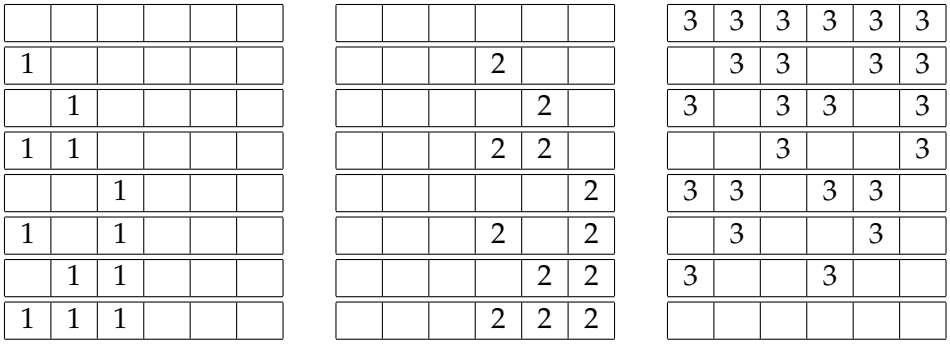


Figure 3.2 Pieces of the USP in Figure 3.1.

only care whether each cell has a j or not. A reimagining of the above USP with respect to these ideas is shown in Figure 3.2.

So while each π_j is defined as a permutation that moves around entire rows of the USP, a much better perspective is that it is merely reordering the rows of the jth grid above.

Imagine breaking the USP into three separate grids with all occurrences of j in each one, just as shown above. Each π_j moves around the rows of one of the grids independently of the others, and then the three grids are superimposed. The selection of u, i is the same as selecting a row and column respectively of the resulting table. Satisfying $(\pi_j(u))_i = j$ is equivalent to the grid having a j in the selected position. The USP rule's constraint that at most one of $(\pi_j(u))_i = j$ is true means there is no overlap between the three permuted rows. The negation of the USP rule, that at least two of $(\pi_j(u))_i = j$ are true, which is the same language in the original definition (Definition 4), refers to the case where two of the permuted rows do overlap. So we can imagine each row of each grid as a jigsaw piece which must be fit with the others in a nonoverlapping fashion. We refer to these overlaps as *collisions*.

The definition for a USP states that if the USP rule is satisfied for all positions, we must have $\pi_1 = \pi_2 = \pi_3$. That is, all of the rows are permuted in the same fashion, and the resulting table is merely a reordering of the rows of the original one. If we imagine a USP candidate as a puzzle to be solved under the constraints of the USP rule, then a USP is one in which the only solution is the original grid that arises from $\pi_1 = \pi_2 = \pi_3$. This is the motivation for the name *uniquely solvable puzzle*.

As an example of working with these conditions, we give a technical demonstration that the grid shown above is in fact a USP in this frame-

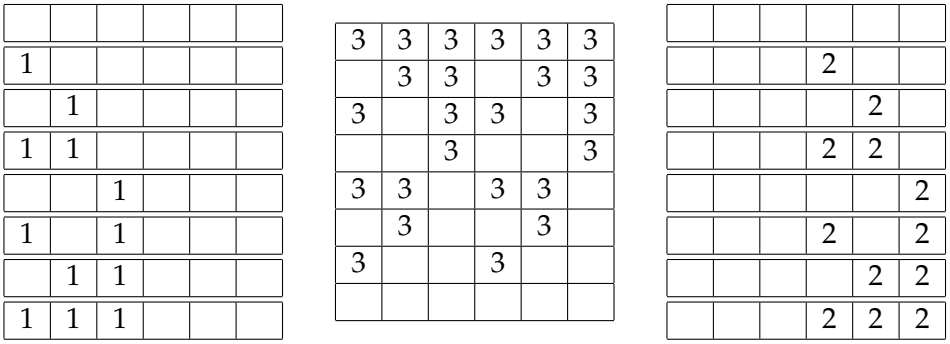


Figure 3.3 Pieces of the USP in Figure 3.1 with 3s placed.

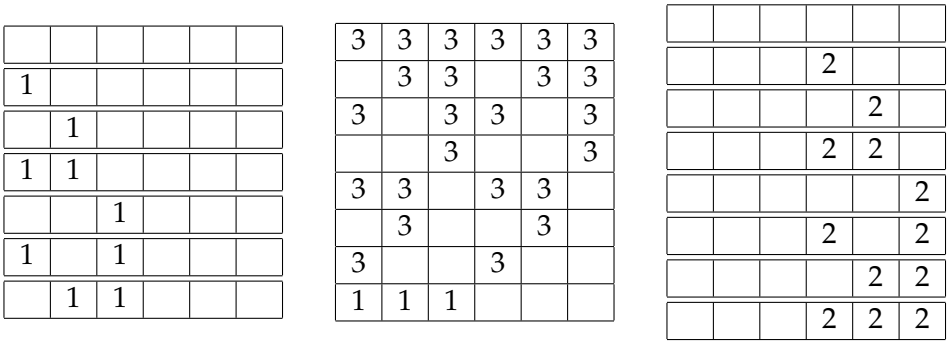


Figure 3.4 Pieces of the USP in Figure 3.1 with 3s and a row of 1s placed.

work. We first assume without loss of generality that $\pi_3 = 1$, the identity, because considering each triple (π_1, π_2, π_3) is equivalent to considering $(\pi_1\rho, \pi_2\rho, \pi_3\rho)$ for some $\rho \in \mathrm{Sym}(U)$. Choose $\rho = \pi_3^{-1}$. $\pi_1\rho, \pi_2\rho$ are still arbitrary permutations while $\pi_3\rho$ is forced to be 1. So this leaves us in the state of Figure 3.3.

The intent of Figure 3.3 is to show the resulting superimposed table, currently incomplete, in the center, with the remaining pieces to be fit in on the two sides. We have $\pi_3 = 1$, so we know where all the 3s go in the table already, and these have been inserted in the center table.

Consider now how we are to fit the row with three 1s into the table. Because all rows of the table contain a 3 in one of the first three columns except the last, attempting to fit the row with three 1s into anywhere else would cause a collision. So we know that row must be assigned to the last position. This can be written into the resulting table, as shown in Figure 3.4.

3	3	3	3	3	3
1	3	3		3	3
3	1	3	3		3
1	1	3			3
3	3	1	3	3	
1	3	1		3	
3	1	1	3		
1	1	1			

			2		
				2	
			2	2	
					2
			2		2
				2	2
			2	2	2

Figure 3.5 Pieces of the USP in Figure 3.1 with 3s and 1s placed.

Now consider any row with two 1s. Again, we find there is only one position that it can go in, since the last row has been filled. We similarly find afterwards that each of the rows with one 1 has only one position left in what remains, giving us the state shown in Figure 3.5.

Since the structure of the pieces with 2s are the same as those with 1s, just in a different set of columns, it should also be clear that they can be uniquely assigned. This shows the grid does in fact have a unique solution under the USP rule, and so it is a USP.

We make one additional note about the case in which two pieces, or rows, are identical, meaning they have the same number and configuration. In this case, we can have that π_j be the permutation only swapping those two rows, which causes no change to the grid, and then having the other two permutations be the identity. This results in a grid matching the original USP, meaning no collisions, and it also does not satisfy $\pi_1 = \pi_2 = \pi_3$. So this can never be a valid USP. The analogy is to a jigsaw puzzle in which two pieces are completely indistinguishable. Such a puzzle can be said not to have a unique solution either.

Recall that we defined the notion of a USP for the eventual purpose of finding groups and subsets satisfying the triple product property. However, with respect to this goal, USPs themselves are just a preliminary concept. To actually obtain groups and triple product property subsets we instead need something called a strong USP, which we define now.

Definition 6. *Let U be a USP candidate and let (π_1, π_2, π_3) be three permutations in $Sym(U)$ which act on the rows of U. Let $u \in U, i \in \{1, 2, \ldots, k\}$ refer to a position in the grid U. We say (π_1, π_2, π_3) satisfy the* strong USP rule *for this*

position if the number of $j \in \{1,2,3\}$ *satisfying* $(\pi_j(u))_i = j$ *is 0, 1, or 3 (i.e., not 2).*

A USP candidate U is a strong uniquely solvable puzzle (USP) *if the only triples of permutations* (π_1, π_2, π_3) *that satisfy the strong USP rule for all positions are those with* $\pi_1 = \pi_2 = \pi_3$.

In the jigsaw puzzle framework shown above, this means that we are now accepting more solutions in which pieces do overlap, provided every space of overlap has all three of the pieces superimposed on each other. Call such an occurrence an *omnicollision*. The USP rule does not allow collisions of any kind; the strong USP rule allows them only if they are omnicollisions.

In the case of the example USP just discussed, there is no column in the grid that contains all three numbers. Therefore it is not possible to have an omnicollision anywhere in the grid, regardless of how the three permutations are chosen. In this case the strong USP rule reduces to the USP rule, so the grid being a USP implies that it is also a strong USP.

The complications of working with strong USPs are made abundantly clear later when we discuss attempts to construct USPs, but one insight into this can be made clear right away. The USP rule states that the number of solutions to $(\pi_j(u))_i = j$ must be one or zero. However there are $k \cdot |U|$ spaces in the grid to fill and $k \cdot |U|$ numbers in the jigsaw pieces to place. So by the pigeonhole principle, one number has to be assigned to every space, and therefore there can be no blank spaces in the superimposed grid. This observation is often quite helpful when proving a USP candidate is an ordinary USP, as any case that forces a space to be left blank results in a contradiction.

However, for strong USPs, the new rule allows for omnicollisions. That is, it is possible for $(\pi_j(u))_i = j$ to have three solutions. As a result, it is possible for an alternate solution to a strong USP to have blank spaces, if there are an appropriate number of u_i such that $(\pi_j(u))_i = j$ is true for $j = 1, 2, 3$. It is this possibility that presents the greatest challenge in strong USP construction. Not being able to eliminate possible alternate solutions from the fact that they leave spaces blank greatly reduces the availability of methods for proving a USP candidate is a strong USP, and hence for constructing strong USPs in general.

3.2 Application

We now describe the link between this combinatorial problem and the original one of finding groups and triples of subsets, following the methods of Cohn et al. (2005). After the exposition is complete, we give a new result, Theorem 8, that can aid attempts to construct strong USPs.

Fix an integer $m \geq 3$. Suppose we have a strong USP U of width k; recall that U is technically defined as the set of rows. Let H be the set of functions from $U \times \{1, 2, \ldots, k\}$ to the cyclic group of m elements C_m, and view H as an abelian group with an operation of pointwise addition. Intuitively, we can view an element of H as a grid of rows and columns with the same dimensions as the strong USP U. The two differences are that each cell is an element of C_m instead of an element of $\{1, 2, 3\}$, and we have defined a way to add two grids together.

It is worth making some brief remarks about the integer m at this point. For now the reader should be content to treat this as an unspecified parameter. Its application comes up in Theorem 7, which relates USP constructions to ω directly. We further discuss the mysterious m after stating that result.

Recall that $\mathrm{Sym}(U)$ is the set of permutations of the rows of U. We want to consider the action of $\mathrm{Sym}(U)$ on H. Naturally, for a permutation $\pi \in \mathrm{Sym}(U)$ and a grid $h \in H$, π acts on h by permuting the rows of h. This action is formally defined by

$$\pi(h)(u, i) = h(\pi^{-1}(u), i),$$

where $u \in U$ and $i \in \{1, 2, \ldots, k\}$ as usual represent a position in the grid h.

Let the group G be the semidirect product of H with $\mathrm{Sym}(U)$, using the automorphism induced by the action defined above. One way to understand G is that we take any grids with the same dimensions as the USP U, as in H, and allow the group operation to either move rows around or add two grids by superposition.

Now that we have our group G, we also need to define the three subsets S_1, S_2, and S_3 satisfying the triple product property, which is where we use the entries of the USP. To define S_i for $i \in \{1, 2, 3\}$, include those elements $g \in G$ with $g = \pi h$ where $\pi \in \mathrm{Sym}(U), h \in H$, subject to the condition that h's grid entries are nonzero if and only if the corresponding position in the USP U has an i. There are no restrictions on the permutation π.

In Figure 3.6, we show an example of a small USP U on the left next to a couple example elements from H, where $m = 3$ so that grids in H have

3	3	3	3
1	3	2	3
3	1	3	2
1	1	2	2

0	0	0	0
2	0	0	0
0	2	0	0
1	2	0	0

0	0	0	0
0	0	0	0
0	0	0	1
0	0	1	2

Figure 3.6 Small USP (left) and two example elements of the group H.

entries from $\{0, 1, 2\}$. Here, the element of H shown in the center is also a member of S_1, but the element on the right is not a member of S_2, much less S_1 or S_3, because it has a 0 in one of the positions where the USP has a 2. One can see that for this USP, $|S_1| = |S_2| = 2^4$ and $|S_3| = 2^8$.

The constraints U must satisfy to be a strong USP implies that these subsets satisfy the triple product property. This is Proposition 3.5 in Cohn et al. (2005); we reproduce the argument here. It is required to show that if $q_1q_2q_3 = 1$, then $q_1 = q_2 = q_3 = 1$ where $q_i \in Q(S_i)$ for each i. Let $q_i = h_i\pi_i\pi_i'^{-1}h_i'^{-1}$ for $h_i, h_i' \in H$ and $\pi_i, \pi_i' \in \mathrm{Sym}(U)$. The condition reduces to

$$h_1\pi_1\pi_1'^{-1}h_1'^{-1}h_2\pi_2\pi_2'^{-1}h_2'^{-1}h_3\pi_3\pi_3'^{-1}h_3'^{-1} = 1. \tag{3.1}$$

Note that the permutations, each of which are arbitrary, always occur in the form $\pi_i\pi_i'^{-1}$. Let this expression be denoted ρ_i to simplify matters. Notice that the 1 on the right hand side is an element of G, and so it can be broken up into two components using the semidirect product form of G. One component of it is the identity permutation. So by considering only the permutations on the left hand side, we can say that Equation 3.1 implies that $\rho_1\rho_2\rho_3 = 1$ must be satisfied in $\mathrm{Sym}(U)$. This allows us to simplify Equation 3.1 into the following equation in H albeit equipped with the action of $\mathrm{Sym}(U)$:

$$h_1 - h_3' + \rho_1(h_2 - h_1') + \rho_3^{-1}(h_3 - h_2') = 0. \tag{3.2}$$

Here the 0 on the right hand side represents the identity in H, a grid of all 0s. Also, we can treat ρ_1, ρ_3 as arbitrary permutations. The only constraint on them is that $\rho_1\rho_2\rho_3 = 1$, but this has a solution for any choice of ρ_1, ρ_3 if we choose $\rho_2 = \rho_1^{-1}\rho_3^{-1}$.

Recall that to show the triple product property we originally needed to prove $q_1 = q_2 = q_3 = 1$. Since $q_i = h_i\rho_i h_i'^{-1}$, this reduces to needing $\rho_1 = \rho_3 = 1$, which implies $\rho_2 = 1$ because $\rho_2 = \rho_1^{-1}\rho_3^{-1}$, and $h_i = h_i'$ for all $i \in \{1, 2, 3\}$.

The condition that U is a strong USP allows us to show $\rho_1 = \rho_3 = 1$. Supposing there is a solution to Equation 3.2 with $\rho_1 \neq 1$ or $\rho_3 \neq 1$, take

$(\rho_3^{-1}, 1, \rho_1)$ as a triple of permutations and invoke the strong USP rule on them. Because not all three are equal, there must be some violation of the strong USP condition, meaning that some position in the grid u, i for $u \in U, i \in \{1, 2, 3, \ldots, k\}$ has exactly two of the numbers colliding. Without loss of generality, suppose that 1 and 3 collide there but 2 does not, meaning $(\rho_3^{-1}(u))_i = 1$, $(\rho_1(u))_i = 3$, but $u_i \neq 2$.

Recall that $h_j - h'_k$ has nonzero entries exactly in the positions corresponding to j and k in the USP. Thus, because $(\rho_3^{-1}(u))_i = 1$, we have $\rho_3^{-1}(h_3 - h'_2)$ equal to 0 in the grid position corresponding to u_i. Similarly, $\rho_1(h_2 - h'_1)$ is 0 in this position. But $u_i \in \{1, 3\}$, so $h_1 - h'_3$ has a nonzero entry in the given position. Going back to Equation 3.2, we see that the left side has exactly one nonzero term for this position of the grid. Then the final result of the computation of the left hand side of Equation 3.2 has a nonzero entry in the grid, contradicting it being equal to the right hand side which is the grid of all 0s. Thus, no solution with either $\rho_1 \neq 1$ or $\rho_3 \neq 1$ exists, and Equation 3.2 implies that $\rho_3 = \rho_1 = 1$. If we take that equation, cancel ρ_3 and ρ_1 and rearrange the remaining terms, we get the following:

$$(h_1 - h'_1) + (h_2 - h'_2) + (h_3 - h'_3) = 0. \tag{3.3}$$

To finish, we only have left to show that $h_i = h'_i$ for all $i \in \{1, 2, 3\}$. But remember that two elements from S_i and S_j for $i \neq j$ do not have any positions with nonzero entries in common; this is simply by the way in which the S_i were defined. Since no two terms of the form $h_i - h'_i$ have nonzero entries in the same grid position, the only way to get all 0s in the sum $\sum_{i=1}^{3}(h_i - h'_i)$ is for each individual term to be 0. That is, $h_i - h'_i = 0$ for each $i \in \{1, 2, 3\}$, which is what we wanted to show. Thus, S_1, S_2, and S_3 satisfy the triple product property.

The computations involved in determining the complexity of the algorithm that results from a valid strong USP is omitted; the details can be found in Cohn et al. (2005). However we give the final result here. First, we define an important quantity known as the USP capacity, which is the chief measure of the effectiveness of a USP construction is. In Cohn et al. (2005), it is defined in the text after Proposition 3.1; the definition we present here uses a different but equivalent formulation for clarity purposes.

Definition 7. *Let $\mathcal{F} = \{U_1, U_2, U_3, \ldots\}$ be an infinite family of USP candidates, where the USP U_i has width k_i and $|U_i|$ rows. Define $c_i = |U_i|^{1/k_i}$, and let $C = \limsup_{i\to\infty} c_i$. We call C the* capacity *of the family $\mathcal{F}$.*

The USP capacity *is the largest constant C such that there exists an infinite family $\mathcal{F}$ of capacity C all of whose members are USPs. Similarly, the* strong USP

capacity *is the largest constant C such that there exists an infinite family $\mathcal{F}$ of capacity C all of whose members are strong USPs.*

The aim in constructing USPs is to find the highest possible capacity. From this we can infer that in constructing USPs we must minimize the number of columns but maximize the number of rows.

Our work in this section has detailed a way of obtaining a group and three subsets satisfying the triple product property from a strong USP. In this way every infinite family of strong USPs can be associated to a matrix multiplication algorithm for matrices of arbitrary size. The following result from Cohn et al. (2005) uses the group and subsets we constructed above to relate the complexity of this algorithm to the strong USP capacity:

Theorem 7. *Let ω be the exponent of matrix multiplication. If the strong USP capacity is C and $m \geq 3$ is an integer, then*

$$\omega \leq \frac{3(\log m - \log C)}{\log(m-1)}.$$

Here is the reappearance of the parameter m. It turns out that the optimum bound on ω given by Theorem 7 requires different choices of m depending on the strong USP capacity C, with larger capacities requiring smaller m. For example, if we construct a strong USP family showing $C \geq \sqrt{2} \approx 1.414$, as an example of Cohn et al. (2005) does, then the best bound on ω from the theorem is 2.67 obtained from $m = 9$. On the other hand, if $C = \frac{3}{\sqrt[3]{4}} \approx 1.890$, choosing $m = 3$ leads to $\omega \leq 2$, the theoretical lower on ω.

In light of the last example, it is unsurprising that $\frac{3}{\sqrt[3]{4}}$ is the maximum capacity for both USPs and strong USPs, as shown in Cohn et al. (2005). In fact, $\frac{3}{\sqrt[3]{4}}$ is achievable as a capacity for ordinary (not strong) USPs. But it is an open question as to whether the strong USP capacity is equal to $\frac{3}{\sqrt[3]{4}}$, and since it would imply that $\omega = 2$ directly this is the holy grail of USP construction.

The example provided in Figure 3.1 is the case $k = 3$ of an infinite family of strong USPs with width $2k$ and size 2^k which was described in Cohn et al. (2005). We can compute that the capacity of such a family is $\sqrt{2}$. However, this is relatively low as far as USP capacities can go, and we can easily do much better. We discuss progress on this in the following section.

The definition of the capacity allows for a surprising amount of freedom in the amount of rows in a family of USPs. The following result of ours

demonstrates this, although we suspect it has been known in some form to others.

Theorem 8. *Let $\mathcal{F}$ be an infinite family of USP candidates. Let p be any polynomial such that $p(k) > 1$ for all positive integers k. For each $U \in \mathcal{F}$ of width k and size $|U|$, take a subset of the rows of U with at least $\frac{|U|}{p(k)}$ elements and construct from it a new USP candidate $f(U)$, also of width k. Let $\mathcal{F}'$ be the family of all such $f(U)$. Then $\mathcal{F}'$ has the same capacity as $\mathcal{F}$.*

Proof. Let $\mathcal{F} = \{U_1, U_2, U_3, \ldots\}$ and $\mathcal{F}' = \{f(U_1), f(U_2), f(U_3), \ldots\}$. As in Definition 7, define $c_i = |U_i|^{1/k_i}$ where k_i is the width of U_i and $c_i' = |f(U_i)|^{1/k_i}$.

Choose a real a and positive integer d such that $p(k) \leq ak^d$ for all positive integers k. Then $|f(U_i)| \geq \frac{|U_i|}{ak_i^m}$, so

$$c_i' = |f(U_i)|^{1/k_i} \geq \left(\frac{|U_i|}{ak_i^d}\right)^{1/k_i} = |U_i|^{1/k_i} \cdot \frac{1}{a^{1/k_i}} \cdot \frac{1}{k_i^{d/k_i}} = c_i \cdot a^{-1/k_i} \cdot k_i^{-d/k_i}.$$

As $i \to \infty$, note that $k_i \to \infty$, since there are a finite number of USPs of a given width and our family is infinite. Clearly $\lim_{k_i \to \infty} a^{1/k_i} = 1$; we also wish to show that $\lim_{k_i \to \infty} k_i^{d/k_i} = 1$. Take the log and notice that $\ln(k_i^{d/k_i}) = \frac{d \ln k_i}{k_i}$. Because k_i outgrows $\ln k_i$, this goes to 0 as $k_i \to \infty$. Hence $\lim_{k_i \to \infty} k_i^{-d/k_i} = 1$. Applying these results to the equation above, we get that $\limsup_{i \to \infty} c_i' = \limsup_{i \to \infty} c_i \cdot 1$, and so the capacities are equal as desired. □

So in attempting to construct families of USPs, we can actually reduce the number of rows of each member of a family by any polynomial factor without fear of negatively impacting the capacity. This is useful for the purpose of attempting to find constructions. If we have a USP candidate with many more than one solution, striking out a single well-chosen row could easily reduce the number of solutions by half or more. Thus, being able to strike out half of the rows of the candidate could reduce the number of solutions by a tremendous factor.

3.3 Current Progress

This section covers the current progress that has been made in USP construction, mostly found in Alon et al. (2011) and Cohn et al. (2005). As there has not been much work done on USPs, it is a fairly brief section.

We first briefly point out the existence of a slightly varied definition of USPs in Cohn et al. (2005), which they call a local strong USP. Rather than being applied to triples of permutations of all the rows, a local strong USP applies the condition to every possible triple of rows. Obviously, this stronger condition means a local strong USP is also a strong USP, and the converse is not true. Despite this, Proposition 6.3 of Cohn et al. (2005) shows that the capacities for local strong USPs and strong USPs are the same, although the associated local strong USP is much larger. We have chosen not to investigate local strong USPs in great detail for this thesis.

Regarding the progress that has been made on the construction of families of USPs and giving lower bounds on the strong USP capacity, no work beyond that in Cohn et al. (2005) has been found. Their best result on the strong USP capacity is in Proposition 3.8, where they obtained a capacity of $2^{2/3}$, which leads to $\omega < 2.48$ by Theorem 7. Like the USP of Figure 3.1, it is constructed with the restriction of allowing only two different numbers in a column. As a result, no omnicollisions are possible, so the USP rule implies the strong USP rule. However, they also show that if one imposes this restriction on the columns, the capacity of $2^{2/3}$ that they achieved is the best possible. We discuss this observation and extensions of it in the next section.

The construction in Cohn et al. (2005) that obtains the capacity of $2^{2/3}$ works by looking at the solutions in nonnegative integers a, b, c to the equation $a + b + c = 2^k - 1$. The USP constructed from this has width $3k$, with the first k columns having only 1s and 2s, the next k columns only having 2s and 3s, and the last k only 3s and 1s. This gives 2^k possible configurations for each block of k spaces in a row. The 2^k configurations are indexed from 0 to $2^k - 1$ in some fashion, and then for each solution (a, b, c) to $a + b + c = 2^k - 1$ a row of the USP is constructed with the first k spaces having configuration a, the next k spaces having configuration b, and the last k having configuration c. In Cohn et al. (2005) it is proven that this is a strong USP with some earlier results of the paper that we omit here, which interestingly involve the definition of the triple product property.

In Cohn et al. (2005) the authors also point out connections between the combinatorial constructions needed by Coppersmith and Winograd (1990) and USPs. In fact, the result we stated in the previous section that the USP capacity is exactly $\frac{3}{\sqrt[3]{4}}$ is merely a reinterpretation of a construction done in one section of Coppersmith and Winograd (1990). This suggests that whether we are working in a more elementary setting like Strassen or a more sophisticated framework like Cohn and Umans, the combinatorial

constructions at the heart of matrix multiplication algorithms are the same.

While this is the extent of the work done on finding lower bounds for USP construction, there is a known implication from a different open problem to the negation of the conjecture that the strong USP capacity is $\frac{3}{\sqrt[3]{4}}$. This implication is detailed in Alon et al. (2011) and has to do with a combinatorial object known as a sunflower. A k-sunflower is a collection of k sets $S_1, S_2, \ldots, S_k$ that each have the same pairwise intersection. That is, for each $x \in \bigcup_{i=1}^{k} S_i$ either x is in exactly one set or it is in all k of them. The most interesting question relating to sunflowers is, given a family $\mathcal{F}$ of sets, how big does the family have to be to contain a sunflower. The definition was originally proposed in Erdös and Rado (1960). In that paper, they proved the following fundamental theorem about sunflowers.

Theorem 9. *Let $\mathcal{F}$ be a family of sets, each with cardinality s. If $|\mathcal{F}| > (k-1)^s \cdot s!$ then $\mathcal{F}$ contains a k-sunflower.*

The same 1960 paper also presented the following conjecture, which is one of the most studied problems in combinatorics (Alon et al., 2011).

Conjecture 1. *Let $\mathcal{F}$ be a family of sets, each with cardinality s. There exists a constant c_k depending only on k such that if $|\mathcal{F}| \geq c_k^s$, then $\mathcal{F}$ contains a k-sunflower.*

While this conjecture itself is not related to strong USPs directly, Alon et al. (2011) presents a series of increasingly complex generalizations of it. To reproduce all of these here would require restating most of the content in the paper, so instead readers interested in the details are encouraged to check the original work themselves. It suffices to say that the authors propose a variation of the above conjecture, which rephrases the original definition in $\mathbb{Z}_n$, the integers modulo n, where the original could be seen as being about $\mathbb{Z}_2$ since each element is either in a set or not, and also adds in a multicolored aspect. This conjecture implies that the maximum strong USP capacity is $\left(\frac{3}{\sqrt[3]{4}}\right)^{\epsilon}$, where $\epsilon > 0$ is a parameter of the conjecture. The concept of local strong USPs is used in the proof of this implication.

3.4 Left-Right Paradigm

This section presents an attempt to improve the lower bound on the strong USP capacity. Although it failed to yield a better bound on the capacity, the discussion may help to illuminate some possible ideas and approaches that one might try when constructing strong USPs.

3	3	3	1	3
3	3	2	2	2
3	1	1	2	3
3	1	2	3	2
1	3	2	3	1
1	3	3	3	2
1	1	2	2	3
1	1	1	2	2

Figure 3.7 Example USP using Left-Right Paradigm.

			1	
	1	1		
	1			
1				1
1				
1	1			
1	1	1		

3	3	3		3
3	3			
3				3
3			3	
	3		3	
	3	3	3	
				3

		2	2	2
			2	
		2		2
		2		
				2
			2	
			2	2

Figure 3.8 Pieces of the USP in Figure 3.7 with the 3s fixed.

Consider the USP candidate shown in Figure 3.7. It is a special case of a more general construction mechanism that we refer to as the Left-Right Paradigm, which is described later in this section.

Here is a technical argument for why this is a strong USP. We first show it is a USP; due to a certain constraint applied to the construction, this quickly implies that it is also a strong USP.

For showing this puzzle is a USP, we want to show that the only triple of permutations (π_1, π_2, π_3) which assign exactly one number to each space in the grid have $\pi_1 = \pi_2 = \pi_3$. Without loss of generality, we may specify one of the three permutations to be the identity permutation. For this puzzle, we set $\pi_3 = 1$, so that the positions of the 3s are fixed in place. This gives us the state shown in Figure 3.8.

The only numbers that can go in the empty spaces in columns 1 and 2 are 1s. So for rows $2k-1$ and $2k$ where $k \in \{1, 2, 3, 4\}$, π_1 may either leave them in place or swap them, but if it did anything else it would cause

overlap or empty spaces to be left in columns 1 and 2. So we have narrowed π_1 to sixteen possibilities.

Suppose that π_1 swapped rows 5 and 6. The blank spaces not covered by 1s or 3s must then be covered by 2s. So in the superimposed table, rows 5 and 6 would look like such, identical to the USP of Figure 3.7 except with a swap in column 5.

1	3	2	3	2
1	3	3	3	1

Now consider rows 1 and 2. Regardless of whether π_1 swaps the 1s in these rows or leaves them alone, the configuration of blank spaces that would be left means π_2 must assign either $2, \_, 2$ or $\_, \_, \_$ to one of rows 1 or 2.

3	3	3	2	3
3	3	2	1	2

3	3	3	1	3
3	3	2	2	2

However, both of these rows have already been assigned to rows 5 and 6, so this is not allowed. We show the possibilities for rows 1 and 2 below. In the first case, row 2 needs the $2, \_, 2$ configuration, and in the second case row 1 needs the $\_, \_, \_$ configuration. This contradiction shows that π_1 must fix rows 5 and 6. It can be straightforwardly verified from here that this forces all of the other rows to be fixed as well.

However, this only shows it is a USP. We now argue it is a strong USP. Recall that strong USPs allow collisions if they are omnicollisions; that is, we can have all three numbers overlap in one position. But if a column has just two entries, as columns 1 and 2 do in our puzzle, it is impossible for omnicollisions to occur in that column. So by the same reasoning as before, we can narrow π_1 down to 2^4 possibilities, with rows $2k-1$ and $2k$ either staying in place or being swapped for $k = 1, 2, 3, 4$. Now notice that there is no column which contains a 1 and 3 in the pair of rows $2k-1$ and $2k$ for each $k = 1, 2, 3, 4$. This implies that it is impossible for π_1 and π_3 to collide in the last three columns assuming no collision in the first two, and this in turn means no omnicollision is possible in the last three columns either. So the strong USP rule reduces to the USP rule, and because we have already shown this is a USP, it must also be a strong USP.

The above USP is a special case of a particular construction paradigm which we now describe in full generality. Partition the puzzle's columns into two sections, L and R, standing for left and right. In the above example, L is the two left columns and R is the three right ones. The number of

rows in the puzzle is $2^{|R|}$. Have the 2s only appear in R and make the jigsaw pieces (rows) of 2s be all $2^{|R|}$ possible combinations of them in R's columns. In L, whose width is at most R, have the $2^{|L|}$ ways for the other two pieces $1, 3$ to be arranged in the columns of L each appear $2^{|R|-|L|}$ times.

The configurations of the pieces with 1s and 3s in R are what have to be carefully specified to ensure a strong USP. The main constraint to work with is that for no column in R and two rows with the same signature in L can 1 appear in one row and 3 appear in the other. This ensures that the above proof that we have a strong USP works in general, as no collisions between 1s and 3s can occur in R provided none do in L.

The goal is to maximize $|R|$ given a fixed $|L|$. For $|L| = |R|$ this trivially reduces to the construction of width $2k$ and size 2^k that was shown in Cohn et al. (2005). But the construction above achieves $|L| = 2, |R| = 3$, so there is room for improvement. Further refinements were made to this paradigm using observations like Theorem 8, allowing us to instead take the number of rows to be $\binom{|R|}{|R|/2}$ and only using the configurations of 2s that occupied exactly half of R's columns.

Despite the refinements above, and regardless of any others that one may find, the following proposition places a bound on the best possible capacity of any USP family based on this paradigm.

Proposition 1. *Suppose U is a strong USP of width k using the Left-Right paradigm of this section. That is,*

1. *The columns of U are partitioned into sets L and R with $|L| \leq |R|$,*
2. *The entries of L in each row consist of only 1s and 3s,*
3. *If a pair of rows contain the same entries in the columns of L (the same header), then in no column of R is there one row with a 1 and another row with a 3.*

Then $|U| \leq 2^{2k/3}$. Hence the capacity of any family of strong USPs using the Left-Right paradigm is at most $2^{2/3}$.

Proof. For a given USP U, choose a configuration of 1s and 3s occurring in the columns of L, and look at the set S of rows that have this configuration. By the third condition of the Left-Right paradigm, among the rows of S, each column of R either contains only 1s and 2s or only 2s and 3s. Let r_1 of the columns contain 1s and 2s and r_3 of the columns contain 2s and 3s, so $r_1 + r_3 = |R|$. For U to be a USP, no two rows can have the same configuration of 1s, since otherwise π_1 could swap just those two rows.

Since there are 2^{r_1} different ways to assign 1s to the r_1 columns that can contain them, $|S| \leq 2^{r_1}$. In the same manner we have $|S| \leq 2^{r_3}$, so $|S| \leq 2^{\min(r_1,r_3)}$ which is maximized for $r_1 = r_3 = |R|/2$, giving $|S| \leq 2^{|R|/2}$. Summing over all $2^{|L|}$ possible configurations of 1s and 3s in the columns of L, we have at most $2^{|L|} \cdot 2^{|R|/2}$ total rows in the USP.

At the same time, we are not allowed to have two rows with the same configuration of 2s. So the number of rows in the USP is also at most $2^{|R|}$. So we have the following constraints on $|U|$ and k:

$$\log_2 |U| \leq \min(|L| + |R|/2, |R|), \qquad |L| + |R| = k.$$

This implies $\log_2 |U| \leq \min(k - |R|/2, |R|)$. Since the first argument of the min is decreasing in $|R|$ and the second is increasing, the best bound possible is when $k - |R|/2 = |R|$, which gives $|R| = 2k/3$. Then $\log_2 |U| \leq 2k/3$, implying that $|U| \leq 2^{2k/3}$ as desired. □

The proposition above implies that any family of strong USPs which is constructed with this paradigm has a maximum capacity of $2^{2/3}$, which is at best equal to the best known bound on the capacity (Cohn et al., 2005). This means this construction paradigm cannot lead to an improvement. Whether this method can actually achieve the $2^{2/3}$ capacity is not known, as attempts to find a construction of such an infinite family were dropped upon finding the result of Proposition 1.

This result is similar to Corollary 3.9 of Cohn et al. (2005), which states that a USP of width k with at most two different numbers in each column has size at most $2^{2k/3}$. We conjecture that a generalization of both results is possible, where any USP of width k that places restrictions in order to dodge the strong USP rule entirely is never able to have more than $2^{2k/3}$ rows. We formalize this statement below.

Conjecture 2. *Let U be a strong USP of width k. Suppose for any row $u \in U$ and any three permutations (π_1, π_2, π_3) of the rows of U such that $(\pi_j(u))_i = j$ for some i and all three $j \in \{1, 2, 3\}$ (that is, u_i gives a violation of the USP rule but not the strong USP rule), there exists an i' such that $(\pi_j(u))_{i'} = j$ for exactly two $j \in \{1, 2, 3\}$ (that is, $u_{i'}$ gives a violation of the strong USP rule). Then $|U| \leq 2^{2k/3}$.*

Proving this conjecture is obviously not of the greatest importance in the long run of USP construction, but its statement does provide a guide for what to avoid when constructing USPs.

3.5 SET Paradigm

This section outlines a second attempt to construct a family of USPs with capacity better than $2^{2/3}$, culminating in Conjecture 3 which suggests that this paradigm might be able to achieve a capacity of $\sqrt{3}$. If achieved, this not only beats the current best known capacity but also gives an algorithm faster than Coppersmith and Winograd.

The card game SET is a simple card game whose rules have a particularly elegant phrasing in the language of mathematics. Most of the content of this section dealing directly with SET is drawn from the survey in Davis and Maclagan (2003). A SET deck contains 81 cards, each corresponding to a different vector in $\mathbb{F}_3^4$, where $\mathbb{F}_3 = \{0, 1, 2\}$ is the field of three elements. A SET is a set of three vectors whose sum is 0, meaning that for each component all three vectors have all the same or all pairwise distinct values. This means that three vectors do not form a set if and only if there is a component with two of one value and one of the other, which bears a strong resemblance to the strong USP rule. For USP construction, the question we are interested in is what is the maximum number of cards one can have without any three forming a SET.

This question can be generalized to $\mathbb{F}_3^k$ and is known as the *cap-set problem* (Gowers, 2011; Tao, 2007). A *cap* C is a subset of $\mathbb{F}_3^k$ that does not contain three points x, y, z for which $x + y + z = \mathbf{0}$, where $\mathbf{0}$ is the point with all coordinates 0.

Let a_k be the maximum size of a cap in $\mathbb{F}_3^k$. the first few values of this sequence are 2, 4, 9, 20, 45 and 112 (Davis and Maclagan, 2003; Potechin, 2008). In general we can obtain $2a_k \leq a_{k+1} \leq \frac{1+3a_k}{1+\frac{a_k}{3^{k-1}}}$. The bound on the left comes from the simple idea of taking the maximum cap in $\mathbb{F}_3^k$ and constructing two copies of it in $\mathbb{F}_3^{k+1}$, one with value 0 in the $(k+1)$st component and the other with value 1. This gives us a cap in $\mathbb{F}_3^{k+1}$ of size $2a_k$. The bound on the right is a result of Proposition 6 in Davis and Maclagan (2003). We are particularly interested in how fast the a_k sequence grows, which we capture in the following definition.

Definition 8. *The* solidity $\sigma(C)$ *of a cap* C *in* $\mathbb{F}_3^k$ *is defined to be* $|C|^{1/k}$. *If we let* $\sigma_k = a_k^{1/k}$, *which gives the maximum solidity of a cap in* $\mathbb{F}_3^k$, *the* asymptotic solidity σ *is the maximum solidity achievable by caps in* $\mathbb{F}_3^k$, *defined as* $\sup_k \sigma_k$.

As proven in Proposition 10 of Davis and Maclagan (2003), this supremum turns out to be equal to the limit $\lim_{k\to\infty} \sigma_k$, which is shown to exist in the same result. We postpone current results about σ until the end of

0	0	2		1	2	3	1	2	3	3	1	2
0	1	0		1	2	3	2	3	1	1	2	3
0	2	2		1	2	3	3	1	2	3	1	2
1	0	0		2	3	1	1	2	3	1	2	3
1	1	1	$\mapsto$	2	3	1	2	3	1	2	3	1
1	2	0		2	3	1	3	1	2	1	2	3
2	0	2		3	1	2	1	2	3	3	1	2
2	1	0		3	1	2	2	3	1	1	2	3
2	2	2		3	1	2	3	1	2	3	1	2

Figure 3.9 Example cap C of $\mathbb{F}_3^3$ (left) and associated USP $F(C)$ (right).

this section, when its significance becomes clear. However, it is clear that $2 \leq \sigma \leq 3$ from the definition and what we know about the a_k sequence.

We now show how to utilize caps of $\mathbb{F}_3^k$ for USP construction. Let C be a cap, and create a grid with k columns and $|C|$ rows so that each point p of C corresponds to a row, with one coordinate of p appearing in each column in order from left to right. Then construct a USP candidate $F(C)$ with $3k$ columns and $|C|$ rows by taking the grid and replacing each 0 with $1, 2, 3$; each 1 with $2, 3, 1$; and each 2 with $3, 1, 2$.

As an example, let C be a cap of $\mathbb{F}_3^3$ containing the following nine points:

$$(0,0,2), (0,1,0), (0,2,2), (1,0,0), (1,1,1), (1,2,0), (2,0,2), (2,1,0), (2,2,2).$$

In Figure 3.9, we show the results of computing the USP candidate $F(C)$ for this cap, with the grid of points in C on the left and the USP candidate $F(C)$ on the right. The double borders between some columns in the visualization of $F(C)$ are shown for clarity and are not an inherent part of the USP itself.

Lemma 1. *For any cap C in $\mathbb{F}_3^k$, $F(C)$ is a strong USP.*

Proof. Suppose some triple of permutations (π_1, π_2, π_3) existed which are not all equal but satisfy the strong USP rule. Let x be a row such that $\pi_1^{-1}(x), \pi_2^{-1}(x), \pi_3^{-1}(x)$ are not all the same. Let p_i be the point in the cap corresponding to the USP row $\pi_i^{-1}(x)$, meaning that p_1, p_2, p_3 are the three points corresponding to the rows that π_1, π_2, π_3 superimposed on each other. It is possible two of p_1, p_2, p_3 are equal, but all three cannot be by our choice of x. If all three are distinct, then by the definition of a cap they are not all collinear.

In either case, there exists an index i such that $(p_1)_i, (p_2)_i, (p_3)_i$ have two values equal and the third value different. If $(p_1)_i = (p_2)_i$, then by the way in which we defined F, one of the three columns $3i-2, 3i-1, 3i$ has a 3 collide with a 1 or 2, while it is impossible for 1s and 2s to collide in these columns when $(p_1)_i = (p_2)_i$. This is a contradiction of the strong USP condition. The other cases are handled in exactly the same manner, so (π_1, π_2, π_3) could not have existed. □

Theorem 10. *The strong USP capacity is at least $\sigma^{1/3}$, where σ is the asymptotic solidity.*

Proof. Let $\sigma_{k_1}, \sigma_{k_2}, \ldots$ be an increasing subsequence of $\sigma_1, \sigma_2, \ldots$ such that the limit of this sequence is σ. For each k_i, take a maximal cap C_i in $\mathbb{F}_3^{k_i}$ and construct $F(C_i)$. By Lemma 1 this is a strong USP with $3k_i$ columns and $(\sigma_{k_i}^{1/3})^{3k_i}$ rows. If we define the c_i sequence as in Definition 7, we have $c_i = \sigma_{k_i}^{1/3}$. Since $\sigma^{1/3} - \sigma_{k_i}^{1/3}$ tends to 0, the c_i sequence has limit $\sigma^{1/3}$, so this is the capacity of the family $F(C_1), F(C_2), \ldots$, as desired. □

Notice that the best possible bound on the capacity obtainable by Theorem 10 is $3^{1/3}$ in the case that $\sigma = 3$. This is less than the best known capacity of $2^{2/3}$, so this specific method of constructing USPs is not particularly effective. However, it is a good example of a strong USP family constructed without specifically dodging the constraints of the strong USP rule through restrictions to what numbers can appear in some columns.

Additionally, we also have a refinement of this construction. The USP candidate $G(C)$ is constructed by taking $F(C)$ and deleting every third column, starting from the third column from the left. Figure 3.10 shows an example of a USP candidate $G(C)$, using the same cap C in $\mathbb{F}_3^3$ as was used in the previous example of Figure 3.9.

Lemma 2. *For any cap C in $\mathbb{F}_3^k$, $G(C)$ is a USP.*

Proof. Suppose a triple of permutations π_1, π_2, π_3 existed violating the USP rule. Follow the method of Lemma 1 to obtain three points p_1, p_2, p_3 in the cap C and an index i such that $(p_1)_i, (p_2)_i, (p_3)_i$ have two values equal and the third value different. Since all of our definitions have been symmetric with respect to the 1s, 2s, and 3s of the USP, without loss of generality, we take $(p_1)_i, (p_2)_i$ to be equal and $(p_3)_i$ to be the odd one out. The six possible cases are summarized in Table 3.1. In every case we either obtain a blank space in the superimposed grid, part of which is shown in the last two columns, or a collision. Both are forbidden in an ordinary USP.

□

0	0	2
0	1	0
0	2	2
1	0	0
1	1	1
1	2	0
2	0	2
2	1	0
2	2	2

$\mapsto$

1	2	1	2	3	1
1	2	2	3	1	2
1	2	3	1	3	1
2	3	1	2	1	2
2	3	2	3	2	3
2	3	3	1	1	2
3	1	1	2	3	1
3	1	2	3	1	2
3	1	3	1	3	1

Figure 3.10 Example cap C of $\mathbb{F}_3^3$ (left) and associated USP $G(C)$ (right).

$(p_1)_i$	$(p_2)_i$	$(p_3)_i$	Col. $2i-1$	Col. $2i$
0	0	1	1	23
0	0	2	13	2
1	1	0	2	
1	1	2	23	
2	2	0		1
2	2	1		13

Table 3.1 Summary of case analysis needed in the proof of Lemma 2.

Unfortunately, $G(C)$ does not typically result in strong USPs. For the example above, there exists a second solution shown in the grid below:

1	2	1	2	3	1
1			X	1	2
1	2	3	1	3	1
	X	1		1	2
2	3	2	3	2	3
	3	X		1	2
3	1	1	2	3	1
X			3	1	2
3	1	3	1	3	1

Only rows 2, 4, 6, 8 have been moved around. An X represents a space with an omnicollision, where 1, 2, and 3 have all been superimposed. However, one can check via a computer that this is the only other solution. So

if we delete one of rows 2, 4, 6, and 8 from the above example, we obtain a strong USP. Theorem 8 allows us to make such deletions in limited amounts in order to obtain strong USPs without any loss in the capacity of the family as a whole. Our conjecture is that we can find a systematic and general method to take a USP candidate $G(C)$ and delete an appropriately small number of rows from it in order to make it into a strong USP, which would immediately imply the following result.

Conjecture 3. *The strong USP capacity is at least $\sigma^{1/2}$, where σ is the asymptotic solidity.*

If σ were somehow shown to be equal to 3, this conjecture would result in a capacity of $\sqrt{3}$. This not only beats the current best known bound for strong USP capacity but also obtains from Theorem 7 an $O(n^{2.286})$ algorithm, which beats the one proposed by Coppersmith and Winograd. Of course, two substantial open questions, namely Conjecture 3 and $\sigma = 3$, stand in the way of obtaining this algorithm, and it would not lead to a proof that $\omega = 2$.

The best known bounds on σ, which have not changed since the writing of Davis and Maclagan (2003), are $2.217 < \sigma \leq 3$. The lower bound was achieved in Edel (2004) with a cap using $k = 480$, which was only available as a preprint when Davis and Maclagan (2003) was published. In a blog post, Fields medalist Terence Tao expresses the opinion that σ is probably equal to 3, although he acknowledges the existence of a dissenting opinion (Tao, 2007). As discussed in Gowers (2011), most current research on the problem targets the limiting behavior of the a_k sequence itself, rather than σ. For a long time the best known upper bound was $a_k = O\left(\frac{3^k}{k}\right)$ using Roth's theorem and Fourier analysis. Very recently, Bateman and Katz (2012) improved this bound to $O\left(\frac{3^k}{k^{1+\epsilon}}\right)$ for some fixed universal ϵ. However there has not been an improvement on the lower bound of σ since Davis and Maclagan (2003).

Bibliography

Alon, Noga, Amir Shpilka, and Christopher Umans. 2011. On sunflowers and matrix multiplication. *ECCC TR11-067* 18.

Bateman, Michael, and Nets Katz. 2012. New bounds on cap sets. *J Amer Math Soc* 25:585–613.

Bierbrauer, Juergen, and Yves Edel. 2002. Bounds on affine caps. *Journal of Combinatorial Designs* 10(2).

Bowen, Richard Strong, Bo Chen, Hendrik Orem, and Martijn van Schaardenburg. 2009. Group-theoretic partial matrix multiplication. `arXiv:cs.CC/0902.2407v1`.

Brin, Sergey, and Larry Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems* 30:110–117.

Bürgisser, Peter, Michael Clausen, and M. Amin Shokrollahi. 1997. *Algebraic Complexity Theory*, *Grundlehren der mathematischen Wissenschaften*, vol. 315. Springer-Verlag.

Cohn, Henry, Robert Kleinberg, Balasz Szegedy, and Christopher Umans. 2005. Group-theoretic algorithms for matrix multiplication. *Proceedings of the 46th Annual Symposium on Foundations of Computer Science* 379–388.

Cohn, Henry, and Christopher Umans. 2003. A group-theoretic approach to fast matrix multiplication. *Proceedings of the 44th Annual Symposium on Foundations of Computer Science* 438–449.

Coppersmith, Don, and Shmuel Winograd. 1990. Matrix multiplication via arithmetic progressions. *J Symbolic Computation* 9:251–280.

Davis, Ben, and Diane Maclagan. 2003. The card game SET. *The Mathematical Intelligencer* 25(3):33–40.

Diaconis, Persi. 1988. Group representations in probability and statistics.

Edel, Yves. 2004. Extensions of generalized product caps. *Designs, Codes and Cryptography* 31:5–14.

Erdös, Paul, and Richard Rado. 1960. Intersection theorems for systems of sets. *J London Math Soc* 35:85–90.

Gonzalez-Sanches, Jon, Laureano Gonzalez-Vega, Alejandro Pinera-Nicolas, Irene Polo-Blanco, Jorge Caravantes, and Ignacio F. Rua. 2009. Analyzing group based matrix multiplication algorithms. *Proceedings of the 2009 International Symposium on Symbolic and Algebraic Computation (ISSAC)* 159–166.

Gowers, Tim. 2011. What is difficult about the cap-set problem. Weblog entry. URL http://gowers.wordpress.com/2011/01/11/what-is-difficult-about-the-cap-set-problem/. Accessed 29 March 2012.

Holtz, Olga, James Demmel, Ioana Dumitriu, and Robert Kleinberg. 2007. Fast matrix multiplication is stable. *Numer Math* 106:199–204.

James, Gordon, and Martin Liebeck. 1993. *Representations and Characters of Groups*. Cambridge University Press.

Kolda, Tamara G., and Brett W. Bader. 2009. Tensor decompositions and applications. *SIAM review* 51(3):455–500.

Pan, Victor. 2001. *Structured Matrices and Polynomials: Unified Superfast Algorithms*. Birkhauser.

Potechin, Aaron. 2008. Maximal caps in $AG(6,3)$. *Des Codes Cryptogr* 46(3):243–259.

Robinson, Sara. 2005. Toward an optimal algorithm for matrix multiplication. *SIAM News* 38(9).

Serre, Jean-Pierre. 1986. *Linear Representations of Finite Groups*. GTM, Springer.

Strassen, Volker. 1969. Gaussian elimination is not optimal. *Numerische Mathematik* 13.

Tao, Terence. 2007. Open question: Best bound for cap sets. Weblog entry. URL http://terrytao.wordpress.com/2007/02/23/open-question-best-bounds-for-cap-sets/. Accessed 29 March 2012.

Williams, Virginia. 2011. Breaking the Coppersmith-Winograd barrier. URL http://www.cs.berkeley.edu/~virgi/matrixmult.pdf. Preprint.