A FAST SEARCH ALGORITHM FOR \(\langle m, m, m \rangle\) TRIPLE PRODUCT PROPERTY TRIPLES AND AN APPLICATION FOR 5 \times 5 MATRIX MULTIPLICATION

SARAH HART\(^1\), IVO HEDTKE\(^2\), MATTHIAS MÜLLER-HANEMANN\(^2\) AND SANDEEP MURTHY\(^1\)

Abstract. We present a new fast search algorithm for \(\langle m, m, m \rangle\) Triple Product Property (TPP) triples as defined by Cohn and Umans in 2003. The new algorithm achieves a speed-up factor of 40 up to 194 in comparison to the best known search algorithm. With a parallelized version of the new algorithm we are able to search for TPP triples in groups up to order 55.

As an application we identify a list of groups that would realize 5 \times 5 matrix multiplication with under 100 resp. 125 scalar multiplications (the best known upper bound by Makarov 1987 resp. the trivial upper bound) if they contain a \(\langle 5, 5, 5 \rangle\) TPP triple. With our new algorithm we show that no group can realize 5 \times 5 matrix multiplication better than Makarov's algorithm.

Keywords: Fast Matrix Multiplication, Search Algorithm, Triple Product Property, Group Algebra Rank

AMS Subject Classification (MSC2010): Primary 20-04, 68Q25, Secondary 20D60, 68Q17, 68R05

1. Introduction

1.1. A Very Short History of Fast Matrix Multiplication. The naive algorithm for matrix multiplication is an \(O(n^3)\) algorithm. From Strassen [15] we know that there is an \(O(n^{2.81})\) algorithm for this problem. One of the most famous results is an \(O(n^{2.3755})\) algorithm from Coppersmith and Winograd [4]. Recently, Williams [16] found an algorithm with \(O(n^{2.3727})\) run-time based on the work of Stothers [14]. Let \(M(n)\) denote the number of field operations in characteristic 0 required to multiply two \((n \times n)\) matrices. Then we call \(\omega := \inf \{ r \in \mathbb{R} : M(n) = O(n^r) \}\) the exponent of matrix multiplication. Details about the complexity of matrix multiplication and the exponent \(\omega\) can be found in [1].

1.2. A Very Short History of Small Matrix Multiplication. The naive algorithm uses \(n^3\) multiplications and \(n^3 - n^2\) additions to compute the product of two \(n \times n\) matrices. The famous result \(O(n^{2.81})\) is based on an algorithm that can compute the product of two \(2 \times 2\) matrices with only 7 multiplications. Winograd [17] proved that the minimum number of multiplications required in this case is 7. The exact number \(R(n)\) of required multiplications to compute the product of two \(n \times n\) matrices is not known for \(n > 2\). There are known upper bounds for some cases. Table [1] lists the known upper bounds for \(R(n)\) up to \(n = 5\). Tables for up to \(n = 30\) can be found in [5, Section 4]. Hedtke and Murthy proved in [9, Theorem 7.3] that the group-theoretic framework (discussed in Subsection [1.4]) is not able to produce better bounds for \(R(3)\) and \(R(4)\).

\(^1\)Department of Economics, Mathematics & Statistics, Birkbeck, University of London, Malet Street, London, WC1E 7HX, United Kingdom.
e-mail: s.hart@bbk.ac.uk, s.murthy@ems.bbk.ac.uk

\(^2\)Institute of Computer Science, Martin-Luther-University of Halle-Wittenberg, Von-Seckendorff-Platz 1, 06120 Halle (Saale), Germany.
e-mail: hedtke@informatik.uni-halle.de, muellerh@informatik.uni-halle.de

Manuscript received xx.
We write \( \phi \) and define its size \( G \). Say that a group \( G \) describes the multiplication of \( n \times n \), \( \eta \) holds if and only if \( \eta \) is called the \( \text{TPP} \) if for \( m \) satisfies the so-called \( \text{TPP} \). The main idea of their framework is to embed the matrix multiplication over a ring \( R \) into the group ring \( \mathbb{R}[G] \) of a group \( G \). A group \( G \) admits such an embedding if there are subsets \( S, T, U \) of \( G \) which satisfy the so-called \( \text{TPP} \).

### Definition 1.1 (Rank). [1] Chapter 14 and Definition 14.7] Let \( k \) be a field and \( U, V, W \) finite dimensional \( k \)-vector spaces. Let \( \eta : U \times V \to W \) be a \( k \)-bilinear map. For \( i \in \{1, \ldots, r \} \) let \( f_i \in U^*, g_i \in V^* \) (dual spaces of \( U \) and \( V \) resp. over \( k \)) and \( w_i \in W \) such that

\[
\eta(u, v) = \sum_{i=1}^{r} f_i(u)g_i(v)w_i
\]

for all \( u \in U \) and \( v \in V \). Then \((f_1, g_1, w_1; \ldots; f_r, g_r, w_r)\) is called a \( k \)-bilinear algorithm of length \( r \) for \( \eta \), or simply a \( \text{bilinear algorithm} \) when \( k \) is fixed. The minimal length of all \( \text{bilinear} \) algorithms for \( \eta \) is called the \( \text{rank} R(\eta) \) of \( \eta \). Let \( A \) be a \( k \)-algebra. The \( \text{rank} R(A) \) of \( A \) is defined as the rank of its \( \text{bilinear multiplication map} \).

### Definition 1.2 (Restriction of a bilinear map). [1] Definition 14.27] Let \( \phi : U \times V \to W \) and \( \phi' : U' \times V' \to W' \) be \( k \)-bilinear maps. A \( k \)-\( \text{restriction} \), or simply a \( \text{restriction} \) (when \( k \) is fixed), of \( \phi' \) to \( \phi \) is a triple \((\sigma, \tau, \zeta')\) of linear maps \( \sigma : U \to U' \), \( \tau : V \to V' \) and \( \zeta' : W' \to W \) such that

\[
\phi = \zeta' \circ \phi' \circ (\sigma \times \tau):
\]

\[
\begin{array}{ccc}
U \times V & \xrightarrow{\phi} & W \\
\sigma \times \tau \downarrow & & \zeta' \\
U' \times V' & \xrightarrow{\phi'} & W'
\end{array}
\]

We write \( \phi \leq \phi' \) if there exists a restriction of \( \phi' \) to \( \phi \).

### Table 1. Upper bounds for \( R(2), R(3), R(4) \) and \( R(5) \).

| \( n \times n \) | upper bound for \( R(n) \) algorithm |
|------------------|----------------------------------------|
| 2 x 2            | 7 Strassen [15]                        |
| 3 x 3            | 23 Laderman [10]                       |
| 4 x 4            | 49 Strassen [15]                       |
| 5 x 5            | 100 Makarov [11]                       |

1.4. **The Group-Theoretic Approach of Cohn and Umans.** In 2003 Cohn and Umans introduced in [3] a group-theoretic approach to fast matrix multiplication. The main idea of their framework is to embed the matrix multiplication over a ring \( R \) into the group ring \( \mathbb{R}[G] \) of a group \( G \). A group \( G \) admits such an embedding if there are subsets \( S, T, U \) of \( G \) which satisfy the so-called \( \text{TPP} \).

### Definition 1.3 (right quotient). Let \( G \) be a group and \( X \) be a nonempty subset of \( G \). The \( \text{right quotient} \ Q(X) \) of \( X \) is defined by \( Q(X) := \{ xy^{-1} : x, y \in X \} \).

### Definition 1.4 (Triple Product Property). We say that the nonempty subsets \( S, T, U \) of a group \( G \) satisfy the \( \text{Triple Product Property} \) (TPP) if for \( s \in Q(S), t \in Q(T) \) and \( u \in Q(U) \), \( stu = 1 \) holds if and only if \( s = t = u = 1 \).

Let \( k \) be a field. By \( \langle n, p, m \rangle \) we denote the bilinear map \( k^{n \times p} \times k^{p \times m} \to k^{n \times m}, (A, B) \mapsto AB \) describing the multiplication of \( n \times p \) by \( p \times m \) matrices over \( k \). When \( k \) is fixed, we simply write \( \langle n, p, m \rangle \). \textit{Unless otherwise stated we will only work over} \( k = \mathbb{C} \) in the entire paper. We say that a group \( G \) realizes \( \langle n, p, m \rangle \) if there are subsets \( S, T, U \subseteq G \) of sizes \( |S| = n, |T| = p \) and \( |U| = m \), which satisfy the TPP. In this case we call \( (S, T, U) \) a \( \text{TPP triple} \) of \( G \), and we define its \( \text{size} \) to be \( npm \).
**Definition 1.5** (TPP capacity). We define the **TPP capacity** $\beta(G)$ of a group $G$ as $\beta(G) := \max\{npm : G \text{ realizes } \langle n,p,m \rangle\}$.

Let us now focus on the embedding of the matrix multiplication into $\mathbb{C}[G]$. Let $G$ realize $\langle n,p,m \rangle$ through the subsets $S$, $T$ and $U$. Let $A$ be an $n \times p$ and $B$ be a $p \times m$ matrix. We index the entries of $A$ and $B$ with the elements of $S$, $T$ and $U$ instead of numbers. Now we have

$$(AB)_{s,u} = \sum_{i \in T} A_{s,i} B_{i,u}.$$ 

Cohn and Umans showed that this is the same as the coefficient of $s^{-1}u$ in the product

$$\left(\sum_{s \in S, t \in T} A_{s,t} s^{-1}t\right) \left(\sum_{i \in T, u \in U} B_{i,u} i^{-1}u\right).$$

So we can read off the matrix product from the group ring product by looking at the coefficients of $s^{-1}u$ with $s \in S$ and $u \in U$.

**Definition 1.6** ($r$-character capacity). Let $G$ be a group with the character degrees $\{d_i\}$. We define the $r$-character capacity of $G$ as $D_r(G) := \sum_i d_i^r$.

We write $R(n, p, m)$ for the rank of the bilinear map $\langle n, p, m \rangle$, and $R(n)$ for $R(n, n, n)$. If $G$ realizes $\langle n, p, m \rangle$ then $\langle n, p, m \rangle \leq \mathbb{C}[G]$ (see [3, Theorem 2.3]) by the construction above and therefore $R(n, p, m) \leq R(\mathbb{C}[G]) =: R(G)$:

$$\begin{array}{cc}
\mathbb{C}^{n \times p} \times \mathbb{C}^{p \times m} & \xrightarrow{\text{matrix multiplication}} \mathbb{C}^{n \times m} \\
\text{embedding} \mathbb{C}[G] \times \mathbb{C}[G] & \xrightarrow{\otimes} \mathbb{C}[G] \\
& \xrightarrow{\text{multiplication in } \mathbb{C}[G]} \mathbb{C}[G] \\
& (AB)_{s,u} = \text{coefficient of } s^{-1}u \text{ in } \mathbb{C}[G]
\end{array}$$

From Wedderburn’s structure theorem it follows that $R(G) \leq \sum_i R(d_i)$. The exact value of $R(G)$ is known only in a few cases. So, usually we will work with the upper bound $D_\beta(G) \geq \sum_i R(d_i)$, which follows from the rank $d^3$ of the naive matrix multiplication algorithm for $\langle d, d, d \rangle$. We can now use $\beta(G)$ and $D_r(G)$ to get new bounds for $\omega$:

**Theorem 1.7.** [3, Theorem 4.1] If $G \neq 1$ is a finite group, then $\beta(G) \frac{\hat{\omega}}{2} \leq D_\omega(G)$.

Finally we collect some results to improve the performance of our algorithms in the next sections.

**Lemma 1.8.** [3, Lemma 2.1] Let $(S,T,U)$ be a TPP triple. Then for every permutation $\pi \in \text{Sym}(\{S,T,U\})$ the triple $(\pi(S),\pi(T),\pi(U))$ satisfies the TPP.

**Lemma 1.9.** [12, Observation 2.1] Let $G$ be a group. If $(S,T,U)$ is a TPP triple of $G$, then $(dSa,dTb,dUc)$ is a TPP triple for all $a,b,c,d \in G$, too.

Lemma 1.9 is one of the most useful results about TPP triples. It allows us to restrict the search for TPP triples to sets that satisfy $1 \in S \cap T \cap U$.

**Definition 1.10** (Basic TPP triple). Following Neumann [12], we shall call a TPP triple $(S,T,U)$ with $1 \in S \cap T \cap U$ a **basic** TPP triple.

For that reason, we will assume throughout that every TPP triple is a basic TPP triple.

**Lemma 1.11.** [12, Observation 3.1] If $(S,T,U)$ is a TPP triple, then $|S||(T) + |U| - 1) \leq |G|$, $|T||(|S| + |U| - 1) \leq |G|$ and $|U||(|S| + |T| - 1) \leq |G|$.

**Theorem 1.12.** [9, Theorem 3.1] Three sets $S_1$, $S_2$ and $S_3$ form a TPP triple $(S_1,S_2,S_3)$ if and only if for all $\pi \in \text{Sym}(3)$

$$1 \in S_1 \cap S_2 \cap S_3, \quad Q(S_{\pi_2}) \cap Q(S_{\pi_3}) = 1, \quad \text{and} \quad Q(S_{\pi_1}) \cap Q(S_{\pi_2})Q(S_{\pi_3}) = 1.$$
1.5. The Aim of this Paper. The second and fourth authors of this paper created what we believe are currently the most efficient search algorithms for TPP triples [9]. They also showed that the presented group-theoretic framework is not able to give us new and better algorithms for the multiplication of $3 \times 3$ and $4 \times 4$ matrices over the complex numbers.

To attack the $5 \times 5$ matrix multiplication problem we develop a new efficient search algorithm for $\langle m, m, m \rangle$ (especially $(5,5,5)$) TPP triples. For this special case of TPP triples it is faster than any other search algorithm and it can easily be parallelized to run on a supercomputer.

Even with the new algorithm, it is not feasible simply to test all groups of order less than 100 (best known upper bound for $R(5)$) for $(5,5,5)$ triples. Therefore we develop theoretical methods to reduce the list of candidates that must be checked. We show that the group-theoretic framework cannot give us a new upper bound for $R(5)$.

We will also produce a list of groups that could in theory realize a nontrivial (with less than 125 scalar multiplications) multiplication algorithm for $5 \times 5$ matrices. Additionally we show how it could be possible to construct a matrix multiplication algorithm from a given TPP triple.

2. The Search Algorithm for $(m, m, m)$ TPP Triples

In this section we describe the basic idea and important implementation details for our new fast search algorithm for $(m, m, m)$ triples. The goal of the algorithm is to find possible candidates for TPP triples $(S, T, U)$ using the following necessary and sufficient conditions:

$$1 \in S \cap T \cap U \quad \text{and} \quad Q(S) \cap Q(T) = Q(S) \cap U = Q(T) \cap U = 1. \quad (2)$$

The second condition is a weaker formulation of the known result using $Q(U)$ (in Theorem 1.12), but it is more useful in our algorithm. For each TPP candidate that comes from the algorithm we test if it satisfies the TPP or not (e.g. with a TPP test from [9] Section 4).

Let $G$ be a finite group. Let $n := |G| - 1$. Let $(g_0 := 1_G, g_1, \ldots, g_n)$ be an arbitrary but fixed order of the elements of $G$. We want to find an $(m, m, m)$ TPP triple $(S, T, U)$ (or possible TPP triple candidates) of subsets of $G$. For this, we will represent $S, T$ and $U$ via their basic binary representation:

**Definition 2.1** (binary representation). If $X$ is an arbitrary subset of $G$ we write the *binary representation* $b_X$ of $X$ as an element of $\{0,1\}^{[G]}$, where $(b_X)_\ell = 1$ if and only if $g_\ell \in X$ and $(b_X)_\ell = 0$ otherwise ($0 \leq \ell \leq n$).

Because we only consider basic TPP triples, $(b_S)_0 = (b_T)_0 = (b_U)_0 = 1$, so we only need to consider the binary representations for $1 \leq \ell \leq n$. We call this the *basic binary representation* $b_S^*, b_T^*$ and $b_U^*$. We define $\text{supp}(b_X^*) := \{i : (b_X^*)_i = 1\} = \{i : i > 0, g_i \in X\}$ as the *support* of a basic binary representation $b_X^*$. For example, if $|G| = 8$ and $S = \{1, g_2, g_4, g_7\}$, then

- $b_S^* = (1, 0, 1, 0, 0, 0, 1)$
- $b_T^* = (0, 1, 0, 1, 0, 1)$
- $\text{supp}(b_S^*) = \{2, 4, 7\}$.

We want to sketch the basic idea behind the algorithms with a matrix representation of the possible TPP candidates. This representation is not efficient and will not be used in the algorithms itself. It is only used in this subsection to describe the method. Let $C \in \{0,1\}^{3 \times n}$ denote a matrix representation of a possible TPP candidate. Each row of $C$

$$C = \begin{bmatrix} b_S^* \\ b_T^* \\ b_U^* \end{bmatrix}$$

is the basic binary representation of $S$, $T$, resp. $U$. We can describe the fundamental idea with three steps

(S1) The “moving 1” principle to find the next possible TPP triple candidate after a TPP test for the previous candidates fails.
(S2) The “marking the quotient” routine to realize Equation (2).
(S3) An efficient way to store the matrix $C$ and access its entries.

2.1. The “moving 1” principle. The “moving 1” principle is based on two observations and an idea:

**Observations.**

1. The column sums of $C$ are at most 1.
2. We can restrict the search space for TPP triples with the condition $\min(\text{supp}(b^*_X)) < \min(\text{supp}(b^*_T)) < \min(\text{supp}(b^*_U))$.

**Proof.**

(1) If $M$ is a set with $1_G \in M$ it follows that $M \subseteq Q(M)$. Using Equation (2), we get that $X \cap Y = \{1\}$ for all $X \neq Y \in \{S, T, U\}$. Thus, $\text{supp}(b^*_X) \cap \text{supp}(b^*_T) = \emptyset$ for all $X \neq Y \in \{S, T, U\}$. This proves the statement.

(2) Follows immediately from Lemma 1.8 and the fact that we are looking for TPP triples $(S, T, U)$ with $|S| = |T| = |U|$.

□

The idea of the “moving 1” is as follows: After a TPP test fails we get the next candidate by moving the rightmost 1 in $b^*_U$ one step to the right. If this is not possible, delete the rightmost 1 in $b^*_U$ and move the new rightmost 1. Finally we add the missing 1 to a free spot (remember that the column sums of $C$ are at most 1).

If it is not possible (all 1’s are at the right of $b^*_U$) to move a 1 in $b^*_U$, we delete the whole line $b^*_U$ and move a 1 in $b^*_T$. After this we rebuild a new line $b^*_U$ line from scratch using the two observations above. We do the same with line $b^*_S$ if no more moves in line $b^*_T$ are possible.

**Example.** Let $G$ be group of order 9. We are looking for $\langle 3, 3, 3 \rangle$ TPP triples. The initial configuration of $C \in \{0, 1\}^{3 \times 8}$ would be

$$C = \begin{array}{ccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}$$

which means, that $S = \{1_G, g_1, g_2\}$, $T = \{1_G, g_3, g_4\}$ and $U = \{1_G, g_5, g_6\}$. Now we check, if $(S, T, U)$ satisfies the TPP. If so, we are finished. If not, we generate the next candidate by moving a 1 in $C$:

$$C = \begin{array}{ccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}$$

Now $U = \{1_G, g_5, g_7\}$ and we check the TPP again. The procedure of the “moving 1” continues if the TPP check fails:

In contrast to the example above, the next subsection takes care of $Q(S)$ and $Q(T)$ in Eq. (2).
2.2. The “marking the quotient” routine. To take care of the quotient sets in Eq. (2) we mark the quotient of each row in $C$ in the row itself. This ensures that rows below this row don’t use elements of the quotient sets.

Example. We use the same example as above. We start with $b^*_S = (1, 1, 0, 0, 0, 0, 0, 0)$, which means that

$$C = \begin{pmatrix} 1 & 1 \\ \end{pmatrix}$$

We mark the quotient set $Q(S)$ in line $b^*_S$ with a “q”:

$$C = \begin{pmatrix} 1 & 1 & q \\ \end{pmatrix}$$

So the first possible $b^*_T$ line is

$$C = \begin{pmatrix} 1 & 1 & q \\ 1 & 1 \\ \end{pmatrix}$$

Note that $X \subseteq Q(X)$ for all $X \in \{S, T, U\}$. Thus, we only have to mark the elements in $Q(X) \setminus X =: \tilde{Q}(X)$. Before we can move a 1 in a row $b^*_X$ we have to delete all marks $\tilde{Q}(X)$.

We have to deal with the case, that we found a $b^*_T$ with the “moving 1” principle, but $Q(S) \cap Q(T) \neq \{1\}$: In this situation we have to undo all steps in the process of “marking all elements in $\tilde{Q}(T)$” and we have to find a new $b^*_T$ by moving a 1.

2.3. Efficient Storage of the Basic Binary Representation Matrix. If we use the matrix $C$ to store all necessary information we have to store $3n$ elements and we need exactly 3 tests to check if we can move a 1 to a position $p$: we have to check if $(b^*_S)_p = (b^*_T)_p = (b^*_U)_p = 0$.

We can omit the unnecessary space of $2n$ elements and the unnecessary 2 tests by projecting $C^{3 \times n}$ to a vector $\text{marked} \in \{-2, -1, 0, 1, 2, 3\}^n$:

$$C \rightarrow 1 \cdot b^*_S + (-1) \cdot b^*_Q(S) + 2 \cdot b^*_T + (-2) \cdot b^*_Q(T) + 3 \cdot b^*_U$$

Example. Consider the basic binary representation matrix

$$C = \begin{pmatrix} 1 & 1 & q & q \\ 1 & 1 & q & q \\ \end{pmatrix}$$

The corresponding marked vector is

$$\text{marked} = (1, 1, 2, -1, 2, -1, -2, 3, -2, -2, 3, 0, 0)$$

The check $(b^*_S)_p = (b^*_T)_p = (b^*_U)_p = 0$ can now be done with marked[p] = 0.

2.4. The Search Algorithm. The listing “SearchTPPTripleOfGivenType($G, m$)” shows the pseudo-code for the main function of the search algorithm. The interested reader can get a more detailed version of this pseudo-code, all other pseudo-codes and an implementation in GAP online [8] or via e-mail from the second author.

To test if a given candidate satisfies the TPP, we can use the test algorithms from Hedtke and Murthy [9]. It would also be possible to use a specialized TPP test, because $Q(S)$ and $Q(T)$ are already known and they satisfy Eq. (2).
In this section, we describe an application of the new algorithm. We will show that if a finite group \(G\) admits a \(5,5,5\) triple, then \(R(G) \geq 100\). That is, we cannot improve the current best bound for \(R(5)\) using this particular TPP approach – of course there may be other group-theoretic methods that do yield better bounds. Even with the new algorithm, it is not feasible simply to test all groups of order less than 100 for \(5,5,5\) triples. Therefore we must use theoretical methods to reduce the list of candidates that must be checked. We will also produce a list of groups that could in theory contain a \(5,5,5\) triple for which \(\overline{R}(G) < 125\) (as defined below).

For a finite group \(G\), let \(T(G)\) be the number of irreducible complex characters of \(G\) and \(b(G)\) the largest degree of an irreducible character of \(G\).

We start with two known results.

**Theorem 3.1.** [13, Theorem 6 and Remark 2] Let \(G\) be a group.

1. If \(b(G) = 1\), then \(R(G) = |G|\).
2. If \(b(G) = 2\), then \(R(G) = 2|G| - T(G)\).
3. If \(b(G) \geq 3\), then \(R(G) \geq 2|G| + b(G) - T(G) - 1\).

We write \(\overline{R}(G) := \sum d_i R(d_i)\) for the best known upper bound (follows from Wedderburn’s structure theorem) and \(\underline{R}(G)\) for the best known lower bound (the theorem above) for \(R(G)\).

**Definition 3.2** (C1 and C2 candidates). A group \(G\) that realizes \(5,5,5\) and satisfies \(R(G) < 100\) will be called \(C1\) candidate. A group \(G\) that realizes \(5,5,5\) and satisfies \(\overline{R}(G) < 125\) will be called \(C2\) candidate.

The following is well known, but we include a short proof for ease of reference.

**Lemma 3.3.** If \(G\) is non-abelian, then \(T(G) \leq \frac{5}{3}|G|\). Equality implies that \(|G : Z(G)| = 4\).

**Proof.** If the quotient \(G/Z(G)\) is cyclic, then \(G\) is abelian. Therefore if \(G\) is non-abelian, then \(|G : Z(G)| \geq 4\). Hence \(|Z(G)| \leq \frac{1}{4}|G|\). Now \(T(G)\) is known to equal the number of conjugacy classes of \(G\). For any \(x \in G\), either \(x\) is central or \(|x^G| \geq 2\). The number of conjugacy classes of length at least 2 is \(T(G) - |Z(G)|\). Therefore \(|G| \geq |Z(G)| + 2(T(G) - |Z(G)|)\). This implies \(T(G) \leq \frac{1}{2}(|G| + |Z(G)|) \leq \frac{5}{3}|G|\). Equality is only possible when \(|Z(G)| = \frac{1}{4}|G|\). \(\square\)
Obviously, it is necessary to keep the list of all C1 and C2 candidates as short as possible. To achieve this goal we will develop some common properties of C1 and C2 candidates in this section. We will use them to eliminate as many candidates as possible from the list.

It will be helpful to establish some notation in the particular case where a group has a TPP triple and a subgroup of index 2.

**Definition 3.4.** Let $G$ be a group with a TPP triple $(S, T, U)$, and suppose $H$ is a subgroup of index 2 in $G$. We define $S_0 = S \cap H$, $T_0 = T \cap H$, $U_0 = U \cap H$, $S_1 = S \setminus H$, $T_1 = T \setminus H$ and $U_1 = U \setminus H$.

**Lemma 3.5.** Suppose $G$ realizes $(5, 5, 5)$. If $G$ has a subgroup $H$ of index 2, then $H$ realizes $(3, 3, 3)$.

**Proof.** Suppose $G$ realizes $(5, 5, 5)$ via the TPP triple $(S, T, U)$. If $|S_0| < |S_1|$, then for any $a \in S_1$, replace $S$ by $Sa^{-1}$. This will have the effect of interchanging $S_0$ and $S_1$. Hence we may assume that $|S_0| \geq |S_1|, |T_0| \geq |T_1|$, and $|U_0| \geq |U_1|$. Now $(S_0, T_0, U_0)$ is a TPP triple of $H$, and since each of $S_0$, $T_0$ and $U_0$ has at least 3 elements, clearly $H$ realizes $(3, 3, 3)$. \hfill $\square$

**Lemma 3.6.** Suppose $G$ has a TPP triple $(S, T, U)$. Let $H$ be an abelian subgroup of index 2 in $G$. Then the following hold.

- **a)** $|S_0^{-1}T_0U_0| = |S_0||T_0||U_0|$.
- **b)** $|S_1^{-1}T_1U_1| \geq |S_1||T_1|$.
- **c)** $|S_1^{-1}U_1| = |S_1||U_1|$.
- **d)** $S_0^{-1}T_0U_0 \cap S_1^{-1}T_1U_0 = \emptyset$.
- **e)** $S_0^{-1}T_0U_0 \cap S_1^{-1}U_1T_0 = \emptyset$.
- **f)** $S_1^{-1}T_1U_0 \cap S_1^{-1}U_1T_0 = \emptyset$.

**Proof.** The proof relies almost entirely on the definition of a TPP triple $(S, T, U)$; that if $s \in Q(S)$, $t \in Q(T)$ and $u \notin Q(U)$ with $stu = 1$, then $s = t = u = 1$.

- **a)** The map $(s, t, u) \mapsto s^{-1}tu$ from $S_0 \times T_0 \times U_0$ to $S_0^{-1}T_0U_0$ is clearly surjective. It is also injective: suppose $s^{-1}tu = s^{-1}t\hat{u}$ for some $s, \hat{s} \in S_0$, $t, \hat{t} \in T_0$ and $u, \hat{u} \in U_0$. Then, remembering that $H$ is abelian, we may rearrange to get $(s\hat{s}^{-1})(t\hat{t}^{-1})(u\hat{u}^{-1}) = 1$, forcing (by definition of TPP triple), $s = \hat{s}, t = \hat{t}, u = \hat{u}$. Therefore the map is bijective and $|S_0^{-1}T_0U_0| = |S_0||T_0||U_0|$.

- **b)** The map $(s_1, t_1) \mapsto s_1^{-1}t_1$ from $S_1 \times T_1$ to $S_1^{-1}T_1U_0$ is injective as $s_1^{-1}t_1 = s_1^{-1}t_1 1$, for some $s_1, \hat{s}_1 \in S_1$ and $t_1, \hat{t}_1 \in T_1$, implies $(s_1s_1^{-1})(t_1\hat{t}_1^{-1})(11^{-1}) = 1$, which implies $s_1 = \hat{s}_1$ and $t_1 = \hat{t}_1$. Thus $|S_1^{-1}T_1U_0| = |S_1||T_1|$.

- **c)** The map $(s_1, u_1) \mapsto s_1^{-1}u_1$ from $S_1 \times U_1$ to $S_1^{-1}U_1$ is clearly surjective; it is injective as $s_1^{-1}u_1 = s_1^{-1}\hat{u}_1$ implies $(s_1s_1^{-1})(11^{-1})(u_1\hat{u}_1^{-1}) = 1$ and hence $s_1 = \hat{s}_1$ and $u_1 = \hat{u}_1$. Therefore $|S_1^{-1}U_1| = |S_1||U_1|$.

- **d)** A nonempty intersection $S_0^{-1}T_0U_0 \cap S_1^{-1}T_1U_0 \neq \emptyset$ implies there exist $s_0 \in S_0, t_0 \in T_0, u_0, \hat{u}_0 \in U_0, s_1 \in S_1$ and $t_1 \in T_1$ such that $s_0^{-1}t_0u_0 = s_1^{-1}t_1\hat{u}_0$. But then $t_1^{-1}s_1t_1^{-1}s_0^{-1}t_0u_0\hat{u}_0^{-1} = 1$. Now $t_1^{-1}s_1t_1^{-1}s_0^{-1}t_0u_0\hat{u}_0^{-1} = 1$. Hence we can rearrange to get $(t_0t_1^{-1})(s_0s_1^{-1})(u_0u_0^{-1}) = 1$. Since $(T, S, U)$ is a TPP triple, this implies $s_0 = s_1$, contradicting the fact that $s_0$ and $s_1$ lie in different $H$-cosets. Therefore $S_0^{-1}T_0U_0 \cap S_1^{-1}T_1U_0 = \emptyset$.

- **e)** Suppose for some $s_0 \in S_0, t_0 \in T_0, u_0 \in U_0, s_1 \in S_1$ and $u_1 \in U_1$ we have $s_0^{-1}t_0u_0 = s_1^{-1}u_0t_0$. Then $(s_0s_1^{-1})(u_0u_0^{-1})(t_0t_0^{-1}) = 1$, which implies (by the TPP for $(S, U, T)$ that $s_0 = s_1$, a contradiction. Therefore $S_0^{-1}T_0U_0 \cap S_1^{-1}U_1T_0 = \emptyset$.

- **f)** Suppose for some $s_1, \hat{s}_1 \in S_1, t_0 \in T_0, t_1 \in T_1, u_0 \in U_0$ and $u_1 \in U_1$, we have $s_1^{-1}t_1u_0 = \hat{s}_1^{-1}u_1t_0$. Then $(s_1s_1^{-1})(t_1t_0^{-1})(u_0u_1^{-1}) = 1$, which implies $u_0 = u_1$, a contradiction. Therefore $S_1^{-1}T_1U_0 \cap S_1^{-1}U_1T_0 = \emptyset$. \hfill $\square$
Theorem 3.7. If $G$ realizes $(5,5,5)$ and $|G| \leq 72$, then $G$ has no abelian subgroups of index 2.

Proof. Suppose $G$ has an abelian subgroup $H$ of index 2 and realizes $(5,5,5)$ via the TPP triple $(S,T,U)$. Define $S_0$, $T_0$, $U_0$, $S_1$, $T_1$ and $U_1$ as in Definition 3.3. Then, as in the proof of Lemma 3.9, we may assume $|S_0| \geq 3$, $|T_0| \geq 3$ and $|U_0| \geq 3$. Without loss of generality we may assume that $|S_0| \geq |T_0|$ and $|S_0| \geq |U_0|$. Now since $|G| \leq 72$, we have $|H| \leq 36$. So, from Lemma 3.6 we have

$$36 \geq |H| \geq |S_0^{-1}T_0U_0| = |S_0||T_0||U_0| + |S_1^{-1}U_1T_0| + |S_1^{-1}T_1U_0| \geq |S_0||T_0||U_0| + |S_1||U_1| + |S_1||T_1|.$$  (3)

Using Equation (1) if either $T_0 \geq 4$ or $U_0 \geq 4$, we have $S_0 \geq 4$, which forces $|H| \geq 48$, a contradiction. Thus $|T_0| = |U_0| = 3$. If $S_0 \geq 4$ then we get $|H| \geq 40$, another contradiction. Therefore $|S_0| = |T_0| = |U_0| = 3$, which gives that $|H| \geq 27 + 4 + 4 = 35$, and so $|H| \in \{35,36\}$. If two of $Q(S_0), Q(T_0)$ and $Q(U_0)$ were groups of order 4, then they would generate a subgroup of order 16 in $H$, which is impossible. Therefore, permuting $S,T$ and $U$ if necessary, we may assume that $Q(T_0)$ and $Q(U_0)$ are not subgroups of order 4.

Now consider $S_1^{-1}U_1T_0$. Write $X = S_1^{-1}U_1$. Then $|X| = 4$. If $|XT_0| = 4$, then $XT_0 = X$, and thus $X(T_0) = X$, which implies that $X$ is a union of $(T_0)$-cosets. In particular, $4 = |X|$ divides the order of $(T_0)$. But $T_0$ alone contains 3 elements. Hence $(T_0)$ has order 4. A quick check shows that $Q(T_0) = (T_0)$, contradicting the fact that $Q(T_0)$ is not a subgroup of order 4. We have therefore shown that $|S_1^{-1}U_1T_0| > 4$. A similar argument with $S_1^{-1}T_1U_0$ and $Q(U_0)$ shows that $|S_1^{-1}T_1U_0| > 4$. Substituting back into Equation (3) gives $|H| \geq 27 + 5 + 5 = 37$, a contradiction. Therefore no group of order at most 72 can have both a $(5,5,5)$ triple and an abelian subgroup of index 2.

We are grateful to Peter M. Neumann for pointing out an argument which considerably shortened our proof for the case $|H| = 36$ in the above result.

3.1. C1 Candidates.

Proposition 3.8. If $G$ is a C1 candidate, then $G$ is non-abelian and $45 \leq |G| \leq 72$.

Proof. If $G$ is abelian then $R(G) = |G|$. The maximal size of a TPP triple that $G$ can realize is $|G|$. Therefore $G$ cannot be a C1 candidate. Assume then that $G$ is non-abelian. The fact that $|G| \geq 45$ follows immediately from Lemma 3.11. For the upper bounds, the fact that $T(G) \leq S \frac{1}{2} G$ implies $2|G| - T(G) \geq \frac{1}{2} |G|$ and hence, by Theorem 3.1, $R(G) \geq \frac{1}{2} |G|$. So if $|G| > 72$, then $R(G) > \frac{11 \times 72}{8} = 99$. Hence $G$ is not a C1 candidate. Therefore, if $G$ is a C1 candidate, then $45 \leq |G| \leq 72$.

Theorem 3.9. No group of order 64 is a C1 candidate.

Proof. A GAP calculation of Pospelov’s lower bound on $R(G)$, followed by elimination of any group with an abelian subgroup of index 2, leaves a possible list of seven groups of order 64 that could be C1 candidates. If any of these groups $G$ were to realize a $(5,5,5)$ triple, then any subgroup of order 32 in $G$ would realize a $(3,3,3)$ triple. But a brute-force computer search, similar to that performed by two of the current authors in [9], shows that each of these groups of order 64 has at least one subgroup of order 32 which does not realize $(3,3,3)$. Therefore, no group of order 64 is a C1 candidate.

Theorem 3.10. Table 3 contains all possible C1 candidates.

Proof. By Proposition 3.8 we need only look at groups of order between 45 and 72. A simple GAP program can calculate Pospelov’s lower bound on $R(G)$. Any group for which this bound is greater than 99 can be eliminated. Next, we can eliminate any group with an abelian subgroup
We can use Pospelov’s bound for $R$. Theorem 3.12.

Table 3 contains all possible C2 candidates that are not C1 candidates. (see [9]). Note that we only consider groups that do not realize $\langle 5, 5, 5 \rangle$. 10 APPL. COMPUT. MATH., V.XX, N.XX, 20XX

Runtime.

Proof. We use the same arguments as in the proof of Proposition 3.8. If $G$ is a C2 candidate, then 45 $\leq |G| \leq 90$. $\Box$

3.2. C2 Candidates.

Proposition 3.11. If $G$ is a C2 candidate, then $G$ is non-abelian and 45 $\leq |G| \leq 90$.

Proof. We use the same arguments as in the proof of Proposition 3.8. If $|G| \geq 91$, then $R(G) \geq \frac{11 \times 91}{8} > 125$. Hence $G$ is not a C2 candidate. Therefore if $G$ is a C2 candidate, then 45 $\leq |G| \leq 90$. $\Box$

Theorem 3.12. Table X contains all possible C2 candidates that are not C1 candidates.

Proof. By Proposition 3.11 we can restrict our attention to groups of order between 45 and 90. We can use Pospelov’s bound for $R(G)$ and (for groups of order at most 72) the existence of abelian subgroups of index 2 to eliminate many candidates. After these observations, we look to see if any of the remaining candidates have subgroups of index 2 that do not realize $(3, 3, 3)$. If so, then by Lemma 3.3 the group cannot be a C2 candidate. After this process, 37 groups remain as candidates. Twelve are the existing C1 candidates we already know about. So there are 25 ‘new’ groups here. $\Box$

We note that one of the C2 candidates, $A_5$, is already known ([12, Section 3]) to have a $(5, 5, 5)$ triple so we would not need to check it again computationally.

4. Computations, Tests and Results

4.1. Runtime. We tested our new search algorithm against a specialized version (that only looks for $(m, m, m)$ triples) of the currently best known search algorithm with the test routine TPPTestMurthy (see [3]). Note that we only consider groups that do not realize $(3, 3, 3)$ to show the worst-case runtimes of the searches. Table 4 lists the runtime of the search for $(3, 3, 3)$ TPP triples in non-abelian groups of order up to 26 that satisfy Neumann’s inequality $3(3 + 3 - 1) \leq |G|$. Our algorithm achieves a speed-up of 40 in the worst-case and 194 in the

\[\begin{array}{|c|c|c|c|}
\hline
\text{GAP ID} & \text{structure} & \text{character degree pattern} & R(G) & \overline{R}(G) \\
\hline
[48,3] & C_2 \times C_3 & (1^3, 3^5) & 90 & 118 \\
[48,28] & C_2 \times S_4 = SL(2, 3).C_2 & (1^2, 2^3, 3^1) & 91 & 118 \\
[48,29] & GL(2, 3) & (1^2, 2^3, 3^2, 4^1) & 91 & 118 \\
[48,30] & A_4 \times C_4 & (1^4, 2^2, 3^1) & 88 & 110 \\
[48,31] & C_4 \times C_4 & (1^{12}, 3^4) & 82 & 104 \\
[48,32] & C_2 \times SL(2, 3) & (1^6, 2^6, 3^2) & 84 & 94 \\
[48,33] & SL(2, 3) \times C_2 & (1^6, 2^6, 3^2) & 84 & 94 \\
[48,48] & C_2 \times S_4 & (1^4, 2^2, 3^4) & 88 & 110 \\
[48,49] & C_2 \times A_4 & (1^{12}, 3^4) & 82 & 104 \\
[48,50] & C_2 \times C_3 & (1^3, 3^5) & 90 & 118 \\
[54,10] & C_2 \times (C_2^2 \times C_3) & (1^{18}, 3^4) & 88 & 110 \\
[54,11] & C_2 \times (C_2^2 \times C_3) & (1^{18}, 3^4) & 88 & 110 \\
\hline
\end{array}\]

Table 2. All possible C1 candidates.

1The test were made with GAP 4.6.3 64-bit (compiled with GCC 4.2.1 on OS X 10.8.3 using the included Makefile) on an Intel® Core™ i7-2820QM CPU @ 2.30GHz machine with 8 GB DDR3 RAM @ 1333MHz.
worst-case and 59 in the best-case. We remark that there are cases where the old algorithm is not of particular concern in the context of our problem: the old search algorithm works on $G\langle m\rangle$ and $U\langle m\rangle$ elements to filter TPP triple candidates. The speed-up will be problematically small when $m = 450,450$ candidates and the new algorithm requires no TPP tests at all.

As a worst-case result we get $2^{\approx |m\rangle}$, which is highly dependent on the structure of the groups. But as a worst-case result we get $14$ in the best-case, because the old algorithm is too slow to do a comparison between candidates, and in the worst-case the new algorithm requires no TPP tests at all.

We remark that the speed-up becomes slower when the group becomes larger. However this is not of particular concern in the context of our problem: the old search algorithm works on $S\langle m\rangle$, $T\langle m\rangle$ and $U\langle m\rangle$ and the new algorithm works on $Q(S\langle m\rangle)$, $Q(T\langle m\rangle)$ and $U\langle m\rangle$. So in the best-case the old algorithm uses $|S\langle m\rangle| + |T\langle m\rangle| + |U\langle m\rangle| = 3m$ elements and the new algorithm uses $|Q(S\langle m\rangle)| + |Q(T\langle m\rangle)| + |U\langle m\rangle| = m^2 + m^2 + m$ elements to filter TPP triple candidates. The speed-up will be problematically small when $m^2 \ll |G\rangle$, but you will only look for groups that are near Neumann’s lower bound to get a good matrix multiplication algorithm.

It is not easy to get results about the asymptotic runtime, because that highly depends on the structure of the groups. But as a worst-case result we get

$$\mathcal{O}\left(\frac{|G\rangle!}{m^3(|G\rangle - 3m)!}\right) \times \mathcal{O}\left(m^4 \log m\right) = \mathcal{O}\left(\frac{|G\rangle!m^4 \log m}{m^3(|G\rangle - 3m)!}\right)$$

Table 3. All possible C2 candidates that are not C1 candidates.

| GAP ID | structure | character degree pattern | $R(G)$ | $R(G)$ |
|--------|-----------|-------------------------|--------|--------|
| 52,3   | $C_{13} \times C_{4}$ | $(1^4, 4^3)$ | 100 | 151 |
| 54,5   | $(C_{2} \times C_{4}) \times C_{2}$ | $(16, 2^3, 6^1)$ | 103 | 188 |
| 54,6   | $(C_{2} \times C_{3}) \times C_{2}$ | $(16, 2^3, 6^1)$ | 103 | 188 |
| 54,8   | $(C_{2} \times C_{3}) \times C_{2}$ | $(1^2, 4^2, 4^3)$ | 100 | 122 |
| 55,1   | $C_{11} \times C_{5}$ | $(1^5, 5^2)$ | 107 | 205 |
| 56,11  | $C_{2} \times C_{7}$ | $(17, 7^1)$ | 110 | 265 |
| 57,1   | $C_{19} \times C_{3}$ | $(13, 3^6)$ | 107 | 141 |
| 60,5   | $A_{5}$ | $(1^1, 3^2, 4^1, 5^1)$ | 119 | 196 |
| 60,6   | $C_{3} \times (C_{5} \times C_{4})$ | $(1^{12}, 4^3)$ | 108 | 159 |
| 60,7   | $C_{4} \times C_{4}$ | $(1^4, 2^3, 4^3)$ | 114 | 165 |
| 60,8   | $S_{3} \times D_{10}$ | $(1^4, 2^6, 4^2)$ | 111 | 144 |
| 60,9   | $C_{5} \times A_{4}$ | $(1^5, 3^5)$ | 102 | 130 |
| 63,1   | $C_{7} \times C_{9}$ | $(1^9, 3^6)$ | 113 | 147 |
| 63,3   | $C_{3} \times (C_{7} \times C_{5})$ | $(1^9, 3^6)$ | 113 | 147 |
| 72,16  | $C_{2} \times (C_{2} \times C_{9})$ | $(1^8, 3^6)$ | 122 | 156 |
| 72,47  | $C_{6} \times A_{4}$ | $(1^8, 3^6)$ | 122 | 156 |
| 78,3   | $C_{13} \times S_{3}$ | $(1^2, 3^6, 3^{10})$ | 117 | 117 |
| 80,21  | $C_{5} \times ((C_{4} \times C_{2}) \times C_{2})$ | $(1^40, 2^{10})$ | 110 | 110 |
| 80,22  | $C_{5} \times (C_{4} \times C_{4})$ | $(1^40, 2^{10})$ | 110 | 110 |
| 80,24  | $C_{5} \times (C_{3} \times C_{2})$ | $(1^40, 2^{10})$ | 110 | 110 |
| 80,46  | $C_{10} \times D_{8}$ | $(1^40, 2^{10})$ | 110 | 110 |
| 80,47  | $C_{10} \times Q_{8}$ | $(1^40, 2^{10})$ | 110 | 110 |
| 80,48  | $C_{5} \times ((C_{4} \times C_{2}) \times C_{2})$ | $(1^40, 2^{10})$ | 110 | 110 |
| 88,9   | $C_{11} \times D_{8}$ | $(1^{44}, 2^{11})$ | 121 | 121 |
| 88,10  | $C_{11} \times Q_{8}$ | $(1^{44}, 2^{11})$ | 121 | 121 |
The number of bₙ's can be computed with

\[ \text{# of } bₙ = \sum_{x_1=1}^{\left\lfloor |G|/3 \right\rfloor} \sum_{x_2=x_1+1}^{\left\lfloor (|G|-2)/3 \right\rfloor} \sum_{x_3=x_2+1}^{\left\lfloor (|G|-1)/3 \right\rfloor} \sum_{x_4=x_3+1}^{\left\lfloor |G|/3 \right\rfloor} 1 = \frac{1}{24}(|G|^4 - 6|G|^3 + 11|G|^2 - 6|G|). \]

The number of bₙ's for all groups in the Tables 2 and 3 can be found in Table 5. We implemented the search algorithm with the optional arguments startrow and numberOfRowOneTests to realize a rudimentary parallelization: With an easy script we construct the set of all possible
As the results show, we were not able to find a group $G$ where $|G| = 5 \times 2$ of the authors who showed the same statement for $3 \times 3$ matrix multiplication with less than 100 scalar multiplications with the group-theoretic framework by Cohn and Umans. This continues the results [9, Theorem 7.3] of two of the authors who showed the same statement for $3 \times 3$ and $4 \times 4$ matrix multiplication.

### 5. How to Construct a Matrix Multiplication Algorithm from a TPP Triple?

As the results show, we were not able to find a group $G$ that realizes $(5, 5, 5)$ with $R(G) < 100$. But the groups in the C2 list could realize $(5, 5, 5)$ with less than 125 scalar multiplication, because $R(G) < 125$. This section shows a strategy to search for a nontrivial $5 \times 5$ matrix multiplication algorithm in the C2 list.

Consider the case, that we found a $(5, 5, 5)$ TPP triple $(S, T, U)$ in a group $G$ of the C2 list. We only now that $R(G) < 125$, so we don’t know if this leads to a nontrivial matrix multiplication algorithm. It could require 125 scalar multiplications or more. To construct the algorithm induced by the given TPP triple we have to construct the embeddings $A \mapsto e_A$ and $B \mapsto e_B$ of the matrices $A = [a_{s,t}]$ and $B = [b_{t,u}]$ in $\mathbb{C}[G]$:

$$a_{s,t} \mapsto a_{s,t} \cdot t^{-1}, \quad b_{t,u} \mapsto b_{t,u} \cdot u^{-1}$$

for all $s \in S$, $t \in T$, $u \in U$. (5)

The next step is to apply Wedderburn’s structure theorem:

$$\mathbb{C}[G] \cong \mathbb{C}^{d_1 \times d_1} \times \mathbb{C}^{d_2 \times d_2} \times \cdots \times \mathbb{C}^{d_t \times d_t},$$

where $d_1, \ldots, d_t$ are the character degrees of $G$. The given matrices $A$ and $B$ are now represented by $t$-tuples of matrices $e_A \mapsto (A_1, \ldots, A_t)$ and $e_B \mapsto (B_1, \ldots, B_t)$. The last step is easy: just use the best known algorithms to compute the products $A_i B_i$ or try to make use of the structures (e.g., symmetries, zero entries, ...) in $A_i$ and $B_i$ to find even better algorithms for the small products $A_i B_i$. The back transformation works as in Equation (5) but in the other direction.

Note that it could be possible to use the structure of the zero entries in $A_i$ and $B_i$: There is space for $d_i^2 + \cdots + d_t^2 = |G|$ elements. But we only need space for $|S| \cdot |T|$ resp. $|T| \cdot |U|$ elements.

The key questions for future research are:

- **(Q1)** Are there different embeddings, in the sense that they lead to different structures (pattern of zeros or other types) in the small matrices?
- **(Q2)** Does the number $M(e)$ of multiplications needed to compute the product in $\mathbb{C}[G]$ depend on the embedding $e$?
- **(Q3)** If so, we can bound $R(G)$ by $\min_e M(e)$. How many embeddings $e$ are there and how easy is it to compute $\min_e M(e)$?
We know that $A_5$ realizes $(5, 5, 5)$. There is place for 60 elements in the embedding $e_A \in \mathbb{C}[A_5]$ of a $5 \times 5$ matrix $A$ with 25 elements. The same for $e_R$. So we have to embed at most $|S^{-1}T \cup T^{-1}U| \leq |S^{-1}T| + |T^{-1}U| - 1 \leq 25 + 25 - 1 = 49$ elements into a space of $|A_5| = 60$ elements. Assume that we can fill the lower dimensional parts of the right hand side of \textit{completely}. Thus, only $49-1^2-3^2-3^2-4^2 = 14$ elements of the small matrices in $\mathbb{C}^{5 \times 0}$ are non-zero. Therefore it could be possible, that $A_5$ induces a nontrivial matrix multiplication algorithm: For the first “complete” parts we need $R(1)+2R(3)+R(4) = 96$ scalar multiplications. We have 28 scalar multiplications left to compute the product of $A_5 B_3$ to beat 125 scalar multiplications.

**Example.** Consider the alternating group $A_5$ on five elements. The character degree pattern is $(1^1, 3^2, 4^1, 5^1)$ and so
\[ \mathbb{C}[A_5] \cong \mathbb{C} \times \mathbb{C}^{3 \times 3} \times \mathbb{C}^{3 \times 3} \times \mathbb{C}^{4 \times 4} \times \mathbb{C}^{5 \times 5}. \]

From our point of view there are five open key questions or ideas one could use for future work.

The first two are obviously the $(5, 5, 5)$ search in the C2 list, together with a practicable method to construct a matrix multiplication algorithm out of a given TPP triple. And C1-like searches for $(6, 6, 6)$ matrix multiplication algorithms and higher.

Is it easy and efficient to implement a search algorithm that does use products of quotients sets like in Theorem \ref{thm:search}?

Is there a constructive algorithm for TPP triples of a given type $\langle n, p, m \rangle$?
As far as we know, the smallest example for a non-trivial matrix multiplication realized by the group-theoretic framework by Cohn and Umans is $⟨40, 40, 40⟩$. The group $G = C_3^3 \wr C_2$ realizes $(2n(n-1), 2n(n-1), 2n(n-1))$ with the rank $R(G) = 2|G| - T(G) = 4n^3 - 1/2(n^6 + 3n^5) = \frac{1}{2}n^3(7n^3 - 3)$, see [2] Section 2 for details. Thus, for $n = 5$ it realizes $40 \times 40$ matrix multiplication with 54,500 scalar multiplications. This is way better than the naive matrix multiplication algorithm with $40^3 = 64,000$ scalar multiplications. On the other hand this is not a good result at all: Using $R(40) = R(2^3 \cdot 5) \leq R(2)^3 R(5) \leq 7^3 \cdot 100 = 34300$ we get an even better algorithm. The best known upper bound for the number of scalar multiplications in this case is

$$\frac{n^3 + 12n^2 + 11n}{3} = \frac{40^3 + 12 \cdot 40^2 + 11 \cdot 40}{3} = 27,880$$

by [5] Proposition 2. Maybe our new algorithm can help to find a minimal working example for a non-trivial matrix multiplication algorithm realized with the group-theoretic framework by Cohn and Umans.

REFERENCES

[1] Peter Bürgisser, Michael Clausen, and M. Amin Shokrollahi. Algebraic Complexity Theory, volume 315 of Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]. Springer-Verlag, Berlin, 1997. With the collaboration of Thomas Lickteig.

[2] Henry Cohn, Robert Kleinberg, Balazs Szegedy, and Christopher Umans. Group-theoretic Algorithms for Matrix Multiplication. pages 379–388, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[3] Henry Cohn and Christopher Umans. A Group-theoretic Approach to Fast Matrix Multiplication. pages 348–449, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[4] Don Coppersmith and Shmuel Winograd. Matrix Multiplication via Arithmetic Progressions. J. Symbolic Comput., 9(3):251–280, 1990.

[5] Charles-Éric Drevet, Md. Nazrul Islam, and Éric Schost. Optimization techniques for small matrix multiplication. Theoretical Computer Science, 412(22):219–2236, 2011.

[6] The GAP Group. GAP – Groups, Algorithms, and Programming, Version 4.6.3, 2013.

[7] Jon Gonzalez-Sanchez, Laureano Gonzalez-Vega, Alejandro Piñera Nicolas, Irene Polo-Blanco, Jorge Cara-vantes, and Ignacio F. Rua. Analyzing group based matrix multiplication algorithms. In Proceedings of the 2009 International Symposium on Symbolic and Algebraic Computation, ISSAC ’09, pages 159–166, New York, NY, USA, 2009. ACM.

[8] Sarah Hart, Ivo Hedtke, Matthias Müller-Hannemann and Sandeep Murthy. Online Results and Algorithms of the Search for $(5, 5, 5)$ TPP triples. http://www2.informatik.uni-halle.de/da/hedtke/555/

[9] Ivo Hedtke and Sandeep Murthy. Search and test algorithms for triple product property triples. Groups Complex. Cryptol., 4(1):111–133, 2012.

[10] Julian D. Laderman. A noncommutative algorithm for multiplying $3 \times 3$ matrices using 23 multiplications. Bull. Amer. Math. Soc., 82(1):126–128, 1976.

[11] O. M. Makarov. A non-commutative algorithm for multiplying $5 \times 5$ matrices using one hundred multiplications. U.S.S.R. Comput. Math. Math. Phys., 27(1):205–207, 1987.

[12] Peter M. Neumann. A note on the triple product property for subsets of finite groups. LMS J. Comput. Math., 14:232–237, 2011.

[13] Alexey Pospelov. Group-Theoretic Lower Bounds for the Complexity of Matrix Multiplication. In Theory and Applications of Models of Computation, volume 6648 of Lecture Notes in Comput. Sci., pages 2–13. Springer, Heidelberg, 2011.

[14] Andrew James Stothers. On the Complexity of Matrix Multiplication. PhD thesis, University of Edinburgh, 2010.

[15] Volker Strassen. Gaussian Elimination is not Optimal. Numer. Math., 13:354–356, 1969.

[16] Virginia Vassilevska Williams. Multiplying Matrices Faster Than Coppersmith-Winograd. In Proceedings of the 44th Symposium on Theory of Computing, STOC ’12, pages 887–898, New York, NY, USA, 2012. ACM.

[17] Shmuel Winograd. On Multiplication of $2 \times 2$ Matrices. Linear Algebra and Appl., 4:381–388, 1971.