

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

SARAH HART¹, IVO HEDTKE², MATTHIAS MÜLLER-HANNEMANN² AND SANDEEP MURTHY¹

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 5×5 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 5×5 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 $\mathcal{O}(n^3)$ algorithm. From Strassen [15] we know that there is an $\mathcal{O}(n^{2.81})$ algorithm for this problem. One of the most famous results is an $\mathcal{O}(n^{2.3755})$ algorithm from Coppersmith and Winograd [4]. Recently, Williams [16] found an algorithm with $\mathcal{O}(n^{2.3727})$ 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 $\omega := \inf\{r \in \mathbb{R} : M(n) = \mathcal{O}(n^r)\}$ the exponent of matrix multiplication. Details about the complexity of matrix multiplication and the exponent ω 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 $\mathcal{O}(n^{2.81})$ is based on an algorithm that can compute the product of two 2×2 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 $R(4)$.

¹Department 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

²Institute 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.

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

TABLE 1. Upper bounds for $R(2)$, $R(3)$, $R(4)$ and $R(5)$.

1.3. Bilinear Complexity. Later we will use the concept of bilinear complexity to connect group-theoretic arguments with the complexity of matrix multiplication.

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 $\eta: U \times V \rightarrow W$ be a k -bilinear map. For $i \in \{1, \dots, r\}$ 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 $(f_1, g_1, w_1; \dots; f_r, g_r, w_r)$ is called a k -bilinear algorithm of length r for η , or simply a bilinear algorithm when k is fixed. The minimal length of all bilinear algorithms for η is called the rank $R(\eta)$ of η . Let A be a k -algebra. The rank $R(A)$ of A is defined as the rank of its bilinear multiplication map.

Definition 1.2 (Restriction of a bilinear map). [1, Definition 14.27] Let $\phi: U \times V \rightarrow W$ and $\phi': U' \times V' \rightarrow W'$ be k -bilinear maps. A k -restriction, or simply a restriction (when k is fixed), of ϕ' to ϕ is a triple (σ, τ, ζ') of linear maps $\sigma: U \rightarrow U'$, $\tau: V \rightarrow V'$ and $\zeta': W' \rightarrow W$ such that $\phi = \zeta' \circ \phi' \circ (\sigma \times \tau)$:

$$\begin{array}{ccc} U \times V & \xrightarrow{\phi} & W \\ \sigma \times \tau \downarrow & \circlearrowleft & \uparrow \zeta' \\ U' \times V' & \xrightarrow{\phi'} & W' \end{array}$$

We write $\phi \leq \phi'$ if there exists a restriction of ϕ' to ϕ .

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 $R[G]$ of a group G . A group G admits such an embedding if there are subsets S, T and U of G which satisfy the so-called *Triple Product Property*.

Definition 1.3 (right quotient). Let G be a group and X be a nonempty subset of G . The 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 and U of a group G satisfy the *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 $\langle n, p, m \rangle_k$ we denote the bilinear map $k^{n \times p} \times k^{p \times m} \rightarrow k^{n \times m}$, $(A, B) \mapsto AB$ describing the multiplication of $n \times p$ by $p \times m$ matrices over k . When k is fixed, we simply write $\langle n, p, m \rangle$. 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 TPP triple of G , and we define its size to be npm .

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 3×3 and 4×4 matrices over the complex numbers.

To attack the 5×5 matrix multiplication problem we develop a new efficient search algorithm for $\langle m, m, m \rangle$ (especially $\langle 5, 5, 5 \rangle$) 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 $\langle 5, 5, 5 \rangle$ 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 5×5 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 $\langle m, m, m \rangle$ TPP TRIPLES

In this section we describe the basic idea and important implementation details for our new fast search algorithm for $\langle m, m, m \rangle$ 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 $(g_0 := 1_G, g_1, \dots, g_n)$ be an arbitrary but fixed order of the elements of G . We want to find an $\langle m, m, m \rangle$ 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

$$\begin{aligned} b_S &= (1, 0, 1, 0, 1, 0, 0, 1) \\ b_S^* &= (0, 1, 0, 1, 0, 0, 1) \\ \text{supp}(b_S^*) &= \{2, 4, 7\}. \end{aligned}$$

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

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

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 $\min(\text{supp}(b_S^*)) < \min(\text{supp}(b_T^*)) < \min(\text{supp}(b_U^*))$.*

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, $\text{supp}(b_X^*) \cap \text{supp}(b_Y^*) = \emptyset$ 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.

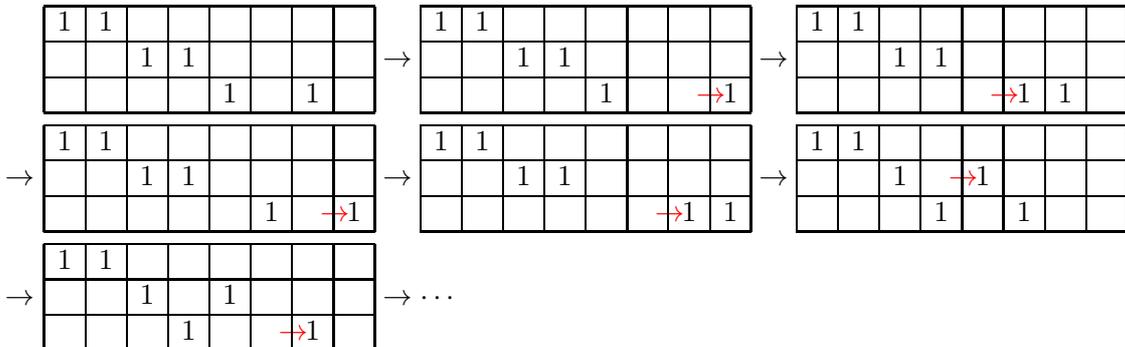
Example. Let G be group of order 9. We are looking for $\langle 3, 3, 3 \rangle$ TPP triples. The initial configuration of $C \in \{0, 1\}^{3 \times 8}$ would be

$$C = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 1 & & & & & & \\ \hline & & 1 & 1 & & & & \\ \hline & & & & 1 & 1 & & \\ \hline \end{array} \quad \begin{array}{l} b_S^* = (1, 1, 0, 0, 0, 0, 0, 0) \\ b_T^* = (0, 0, 1, 1, 0, 0, 0, 0) \\ b_U^* = (0, 0, 0, 0, 1, 1, 0, 0) \end{array}$$

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 :

$$C = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 1 & & & & & & \\ \hline & & 1 & 1 & & & & \\ \hline & & & & 1 & \rightarrow 1 & & \\ \hline \end{array}$$

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

$$C = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 1 & & & & & & \\ \hline & & & & & & & \\ \hline & & & & & & & \\ \hline \end{array}.$$

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

$$C = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 1 & & q & & & & \\ \hline & & & & & & & \\ \hline & & & & & & & \\ \hline \end{array}.$$

So the first possible b_T^* line is

$$C = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 1 & & q & & & & \\ \hline & & 1 & & 1 & & & \\ \hline & & & & & & & \\ \hline \end{array}.$$

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

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 $\bar{Q}(T)$ ” 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 *marked* $\in \{-2, -1, 0, 1, 2, 3\}^n$:

$$C \mapsto 1 \cdot b_S^* + (-1) \cdot b_{\bar{Q}(S)}^* + 2 \cdot b_T^* + (-2) \cdot b_{\bar{Q}(T)}^* + 3 \cdot b_U^*$$

Example. Consider the basic binary representation matrix

$$C = \begin{array}{|c|c|c|c|c|c|c|c|c|c|} \hline 1 & 1 & & q & & q & & & & \\ \hline & & 1 & & 1 & & q & & q & q \\ \hline & & & & & & & 1 & & 1 \\ \hline \end{array}.$$

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 $\text{marked}[p] = 0$.

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).

```

SearchTPPTripleOfGivenType( $G, m$ )


---


for  $i = 1, \dots, m - 1$  do                                     // start with  $S = \{1_G, g_1, g_2, \dots, g_{m-1}\}$ 
|    $\text{marked}[i] := 1$ ;
repeat
|   mark quotient set  $\bar{Q}(S)$  of row  $b_S^*$ ;
|   if it is possible to generate a feasible row  $b_T^*$  from scratch then
|   |   repeat
|   |   |   if it is possible to mark the quotient set  $\bar{Q}(T)$  of  $b_T^*$  without a conflict with  $Q(S)$  then
|   |   |   |   if it is possible to generate a feasible row  $b_U^*$  from scratch then
|   |   |   |   |   repeat
|   |   |   |   |   |   if  $(S, T, U)$  is a TPP triple then                                     // use a test from [9]
|   |   |   |   |   |   |   return  $(S, T, U)$ ;
|   |   |   |   |   |   |   until it is not possible to use the “moving 1” principle for  $b_U^*$  anymore;
|   |   |   |   |   |   |   unmark the quotient set  $\bar{Q}(T)$  of  $b_T^*$ ;
|   |   |   |   |   until it is not possible to use the “moving 1” principle for  $b_T^*$  anymore;
|   |   |   |   unmark the quotient set  $\bar{Q}(S)$  of  $b_S^*$ ;
|   |   until it is not possible to use the “moving 1” principle for  $b_S^*$  anymore;


---



```

3. AN APPLICATION FOR 5×5 MATRIX MULTIPLICATION

In this section, we describe an application of the new algorithm. We will show that if a finite group G admits a $\langle 5, 5, 5 \rangle$ 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 $\langle 5, 5, 5 \rangle$ 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 $\langle 5, 5, 5 \rangle$ triple for which $\underline{R}(G) < 125$ (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 $\bar{R}(G) := \sum_i R(d_i)$ 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 $\langle 5, 5, 5 \rangle$ and satisfies $\underline{R}(G) < 100$ will be called *C1 candidate*. A group G that realizes $\langle 5, 5, 5 \rangle$ and satisfies $\underline{R}(G) < 125$ 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 $T(G) \leq \frac{5}{8}|G|$. 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 $T(G) \leq \frac{1}{2}(|G| + |Z(G)|) \leq \frac{5}{8}|G|$. 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 $\langle 5, 5, 5 \rangle$. If G has a subgroup H of index 2, then H realizes $\langle 3, 3, 3 \rangle$.*

Proof. Suppose G realizes $\langle 5, 5, 5 \rangle$ 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 $\langle 3, 3, 3 \rangle$. \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) $|S_1^{-1}T_1U_0| \geq |S_1||T_1|$;
- 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 $u \in Q(U)$ 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 $s^{-1}tu = \hat{s}^{-1}\hat{t}\hat{u}$ 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 $(\hat{s}s^{-1})(\hat{t}t^{-1})(\hat{u}u^{-1}) = 1$, 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 $S_1^{-1}T_1$ is injective as $s_1^{-1}t_1 = \hat{s}_1^{-1}\hat{t}_1$, for some $s_1, \hat{s}_1 \in S_1$ and $t_1, \hat{t}_1 \in T_1$, implies $(\hat{s}_1s_1^{-1})(\hat{t}_1t_1^{-1})(11^{-1}) = 1$, which implies $s_1 = \hat{s}_1$ and $t_1 = \hat{t}_1$. Thus $|S_1^{-1}T_1| \geq |S_1||T_1|$.
- 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 $s_1^{-1}u_1 = \hat{s}_1^{-1}\hat{u}_1$ implies $(\hat{s}_1s_1^{-1})(\hat{u}_1u_1^{-1})(11^{-1}) = 1$ 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 $t_1^{-1}s_1s_0^{-1}t_0u_0\hat{u}_0^{-1} = 1$. Now $t_1^{-1}s_1$, s_0 and t_0 are all elements of the abelian group H . Therefore we can rearrange to get $(t_0t_1^{-1})(s_1s_0^{-1})(u_0\hat{u}_0^{-1}) = 1$. 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 $s_0 \in S_0$, $t_0, \hat{t}_0 \in T_0$, $u_0 \in U_0$, $s_1 \in S_1$ and $u_1 \in U_1$ we have $s_0^{-1}t_0u_0 = s_1^{-1}u_1\hat{t}_0$. Then $(s_0s_1^{-1})(u_1u_0^{-1})(\hat{t}_0t_0^{-1}) = 1$, 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 $(\hat{s}_1s_1^{-1})(t_1t_0^{-1})(u_0u_1^{-1}) = 1$, which implies $u_0 = u_1$, a contradiction. Therefore $S_1^{-1}T_1U_0 \cap S_1^{-1}U_1T_0 = \emptyset$. \square

Theorem 3.7. *If G realizes $\langle 5, 5, 5 \rangle$ 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 $\langle 5, 5, 5 \rangle$ via the TPP triple (S, T, U) . Define S_0, T_0, U_0, S_1, T_1 and U_1 as in Definition 3.4. Then, as in the proof of Lemma 3.5, 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

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

$$\geq |S_0||T_0||U_0| + |S_1||U_1| + |S_1||T_1|. \quad (4)$$

Using Equation (4) 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 $X\langle T_0 \rangle = X$, which implies that X is a union of $\langle T_0 \rangle$ -cosets. In particular, $4 = |X|$ divides the order of $\langle T_0 \rangle$. But T_0 alone contains 3 elements. Hence $\langle T_0 \rangle$ has order 4. A quick check shows that $Q(T_0) = \langle T_0 \rangle$, 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 $\langle 5, 5, 5 \rangle$ triple and an abelian subgroup of index 2. \square

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 1.11. For the upper bounds, the fact that $T(G) \leq \frac{5}{8}|G|$ implies $2|G| - T(G) \geq \frac{11}{8}|G|$ and hence, by Theorem 3.1, $R(G) \geq \frac{11}{8}|G|$. 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 $45 \leq |G| \leq 72$. \square

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 $\langle 5, 5, 5 \rangle$ triple, then any subgroup of order 32 in G would realize a $\langle 3, 3, 3 \rangle$ 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 $\langle 3, 3, 3 \rangle$. Therefore, no group of order 64 is a C1 candidate. \square

Theorem 3.10. *Table 2 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

GAP ID	structure	character degree pattern	$\underline{R}(G)$	$\overline{R}(G)$
[48,3]	$C_4^2 \rtimes C_3$	$(1^3, 3^5)$	90	118
[48,28]	$C_2.S_4 = \text{SL}(2, 3).C_2$	$(1^2, 2^3, 3^2, 4^1)$	91	118
[48,29]	$\text{GL}(2, 3)$	$(1^2, 2^3, 3^2, 4^1)$	91	118
[48,30]	$A_4 \rtimes C_4$	$(1^4, 2^2, 3^4)$	88	110
[48,31]	$C_4 \times A_4$	$(1^{12}, 3^4)$	82	104
[48,32]	$C_2 \times \text{SL}(2, 3)$	$(1^6, 2^6, 3^2)$	84	94
[48,33]	$\text{SL}(2, 3) \rtimes C_2$	$(1^6, 2^6, 3^2)$	84	94
[48,48]	$C_2 \times S_4$	$(1^4, 2^2, 3^4)$	88	110
[48,49]	$C_2^2 \times A_4$	$(1^{12}, 3^4)$	82	104
[48,50]	$C_2^4 \rtimes C_3$	$(1^3, 3^5)$	90	118
[54,10]	$C_2 \times (C_3^2 \rtimes C_3)$	$(1^{18}, 3^4)$	88	110
[54,11]	$C_2 \times (C_9 \rtimes C_3)$	$(1^{18}, 3^4)$	88	110

TABLE 2. All possible C1 candidates.

of index 2 by Theorem 3.7, and any group of order 64 by Theorem 3.9. This reduces the list to 20 groups. Finally, we observe that if any group of order 48 is a candidate, then any of its subgroups of order 24 must realize a $\langle 3, 3, 3 \rangle$ triple. Another brute-force search on groups of order 24 eliminates ten groups of order 48 from the list. The final list contains ten groups of order 48 and two of order 54. \square

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 $45 \leq |G| \leq 90$. \square

Theorem 3.12. *Table 3 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 $\langle 3, 3, 3 \rangle$. If so, then by Lemma 3.5, 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. \square

We note that one of the C2 candidates, A_5 , is already known ([12, Section 3]) to have a $\langle 5, 5, 5 \rangle$ 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 $\langle m, m, m \rangle$ triples) of the currently best known search algorithm with the test routine `TPPTestMurthy` (see [9]). Note that we only consider groups that do not realize $\langle 3, 3, 3 \rangle$ to show the worst-case runtimes of the searches. Table 4 lists the runtimes¹ of the search for $\langle 3, 3, 3 \rangle$ 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

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 a Intel[®] Core[™] i7-2820QM CPU @ 2.30GHz machine with 8 GB DDR3 RAM @ 1333MHz.

GAP ID	structure	character degree pattern	$\underline{R}(G)$	$\overline{R}(G)$
[52,3]	$C_{13} \rtimes C_4$	$(1^4, 4^3)$	100	151
[54,5]	$(C_3^2 \rtimes C_3) \rtimes C_2$	$(1^6, 2^3, 6^1)$	103	188
[54,6]	$(C_9 \rtimes C_3) \rtimes C_2$	$(1^6, 2^3, 6^1)$	103	188
[54,8]	$(C_3^2 \rtimes C_3) \rtimes C_2$	$(1^2, 2^4, 3^4)$	100	122
[55,1]	$C_{11} \rtimes C_5$	$(1^5, 5^2)$	107	205
[56,11]	$C_2^3 \rtimes C_7$	$(1^7, 7^1)$	110	265
[57,1]	$C_{19} \rtimes C_3$	$(1^3, 3^6)$	107	141
[60,5]	A_5	$(1^1, 3^2, 4^1, 5^1)$	119	196
[60,6]	$C_3 \times (C_5 \rtimes C_4)$	$(1^{12}, 4^3)$	108	159
[60,7]	$C_{15} \rtimes C_4$	$(1^4, 2^2, 4^3)$	114	165
[60,8]	$S_3 \times D_{10}$	$(1^4, 2^6, 4^2)$	111	144
[60,9]	$C_5 \times A_4$	$(1^{15}, 3^5)$	102	130
[63,1]	$C_7 \rtimes C_9$	$(1^9, 3^6)$	113	147
[63,3]	$C_3 \times (C_7 \rtimes C_3)$	$(1^9, 3^6)$	113	147
[72,16]	$C_2 \times (C_2^2 \rtimes C_9)$	$(1^{18}, 3^6)$	122	156
[72,47]	$C_6 \times A_4$	$(1^{18}, 3^6)$	122	156
[78,3]	$C_{13} \times S_3$	$(1^{26}, 2^{13})$	117	117
[80,21]	$C_5 \times ((C_4 \times C_2) \rtimes C_2)$	$(1^{40}, 2^{10})$	110	110
[80,22]	$C_5 \times (C_4 \rtimes C_4)$	$(1^{40}, 2^{10})$	110	110
[80,24]	$C_5 \times (C_8 \rtimes C_2)$	$(1^{40}, 2^{10})$	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]	$C_5 \times ((C_4 \times C_2) \rtimes C_2)$	$(1^{40}, 2^{10})$	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

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

best-case in comparison to the specialized version of Hedtke and Murthy [9]. We are able to shrink the number of candidates that we have to test for the TPP by a factor of 14 in the worst-case and 59 in the best-case. We remark that there are cases where the old algorithm tests 450,450 candidates and the new algorithm requires no TPP tests at all.

We only did tests in the $\langle 3, 3, 3 \rangle$ case, because the old algorithm is too slow to do a comparison like Table 4 for the $\langle 4, 4, 4 \rangle$ case (or higher). The search need only be run in groups that satisfy Neumann's inequality: a group G can only realize $\langle m, m, m \rangle$ if it satisfies $m(2m - 1) \leq |G|$.

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 , T and U and the new algorithm works on $Q(S)$, $Q(T)$ and U . So in the best-case the old algorithm uses $|S|+|T|+|U| = 3m$ elements and the new algorithm uses $|Q(S)|+|Q(T)|+|U| \leq m^2+m^2+m$ elements to filter TPP triple candidates. The speed-up will be problematically small when $m^2 \ll |G|$, 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

$$\underbrace{\mathcal{O}\left(\frac{|G|!}{m!^3(|G|-3m)!}\right)}_{\text{bound for the number of triples that satisfy Eq. (2)}} \times \underbrace{\mathcal{O}(m^4 \log m)}_{\substack{\text{worst-case runtime} \\ \text{for a TPP test with} \\ \text{Thm. 1.12}}} = \mathcal{O}\left(\frac{|G|! m^4 \log m}{m!^3(|G|-3m)!}\right)$$

GAP ID	structure	average* runtime in ms		speed-up	number of TPP tests		search space reduction factor**
		new algo.	old algo.		new algo.	old algo.	
[16,3]	$(C_4 \times C_2) \rtimes C_2$	192	20,133	104	11,595	450,450	38
[16,4]	$C_4 \times C_4$	140	19,481	139	0	450,450	∞
[16,6]	$C_8 \times C_2$	116	19,631	169	0	450,450	∞
[16,7]	D_{16}	241	20,416	84	14,336	450,450	31
[16,8]	QD_{16}	162	20,060	123	9,005	450,450	50
[16,9]	Q_{16}	99	19,250	194	0	450,450	∞
[16,11]	$C_2 \times D_8$	311	20,079	64	19,314	450,450	23
[16,12]	$C_2 \times Q_8$	135	18,667	138	7,628	450,450	59
[16,13]	$(C_4 \times C_2) \rtimes C_2$	201	19,538	97	12,107	450,450	37
[18,1]	D_{18}	658	51,899	78	39,499	1,113,840	28
[18,3]	$C_3 \times S_3$	341	50,360	147	20,134	1,113,840	55
[18,4]	$C_3^2 \times C_2$	646	51,131	79	39,999	1,113,840	27
[20,1]	$C_5 \times C_4$	1,028	119,588	116	54,233	2,441,880	45
[20,3]	$C_5 \times C_4$	1,388	121,702	87	73,971	2,441,880	33
[20,4]	D_{20}	2,033	118,599	58	114,979	2,441,880	21
[22,1]	D_{22}	4,539	241,524	53	248,950	4,883,760	19
[24,1]	$C_3 \times C_8$	5,610	501,854	89	292,340	9,085,230	31
[24,4]	$C_3 \times Q_8$	6,056	498,571	82	303,162	9,085,230	29
[24,5]	$C_4 \times S_3$	7,912	483,640	61	419,556	9,085,230	21
[24,6]	D_{24}	10,711	479,688	44	568,672	9,085,230	15
[24,7]	$C_2 \times (C_3 \times C_4)$	6,623	486,323	73	339,829	9,085,230	26
[24,8]	$(C_6 \times C_2) \rtimes C_2$	8,804	479,182	54	463,453	9,085,230	19
[24,10]	$C_3 \times D_8$	6,540	481,217	73	359,830	9,085,230	25
[24,11]	$C_3 \times Q_8$	5,250	490,716	93	284,001	9,085,230	31
[24,14]	$C_2 \times C_2 \times S_3$	11,555	475,916	41	622,455	9,085,230	14
[26,1]	D_{26}	20,658	832,722	40	1,024,317	15,939,000	15

* The average is taken over 10 runs in which the highest and lowest runtimes are omitted.

** A factor X means that $(\# \text{ TPP test of the new algo.}) \leq \frac{1}{X}(\# \text{ TPP test of the old algo.})$.

TABLE 4. Comparison of the average runtime and number of TPP tests in the search of $\langle 3, 3, 3 \rangle$ TPP triples for the old and the new search algorithm.

as a bound for the runtime of our new algorithm. This is exactly the same bound as for the algorithm in [9]. But as the results in Table 4 show, the real runtime of our new algorithm highly depends on m and the *structure* of the group, whereas the real runtime of the old algorithms seems only to depend on m and the *size* of the group.

4.2. Managing the (Parallel) Computation on a (Super-) Computer. To compute the results (next section) for the search of $\langle 5, 5, 5 \rangle$ TPP triple we used a supercomputer (a cluster with Sun Grid Engine) at the Martin-Luther-University Halle-Wittenberg. The computations (and their management) took several months. The number of b_S^* 's can be computed with

$$\begin{aligned} \# \text{ of } b_S^* \text{'s} &= |\{(x_1, x_2, x_3, x_4) \in \mathbb{N}^4 : 1 \leq x_1 < x_2 < x_3 < x_4 \leq |G|\}| \\ &= \sum_{x_1=1}^{|G|-3} \sum_{x_2=x_1+1}^{|G|-2} \sum_{x_3=x_2+1}^{|G|-1} \sum_{x_4=x_3+1}^{|G|} 1 = \frac{1}{24}(|G|^4 - 6|G|^3 + 11|G|^2 - 6|G|). \end{aligned}$$

The number of b_S^* '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

$ G $	48	52	54	55	56	57
# of b_S^* 's	178,365	249,900	292,825	316,251	341,055	367,290
$ G $	60	63	72	78	80	88
# of b_S^* 's	455,126	557,845	971,635	1,353,275	1,502,501	2,225,895

TABLE 5. Number of b_S^* 's for all groups in the Tables 2 and 3.

b_S^* 's and divide it into subsets of size 1,000 resp. 10,000. Now we start $(\# \text{ of } b_S^* \text{'s})/1,000$ resp. $(\# \text{ of } b_S^* \text{'s})/10,000$ independent jobs on a cluster, each with a different *startrow* that has to check *numberOfRowOneTests* = 1,000 resp. *numberOfRowOneTests* = 10,000 of the b_S^* 's. It is clear that even with an optimized search algorithm this is an immense amount of work. It follows right from that fact, that we dealt with tricks like going from the matrix representation \mathbf{C} to the vector representation *marked* to get a sufficient speed-up to solve the $\langle 5, 5, 5 \rangle$ problem.

4.3. Results. Our search for $\langle 5, 5, 5 \rangle$ TPP triples in all groups of the C1 list showed, that no group can realize 5×5 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 3×3 and 4×4 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 $\langle 5, 5, 5 \rangle$ with $\underline{R}(G) < 100$. But the groups in the C2 list could realize $\langle 5, 5, 5 \rangle$ with less than 125 scalar multiplication, because $\underline{R}(G) < 124$. This section shows a strategy to search for a nontrivial 5×5 matrix multiplication algorithm in the C2 list.

Consider the case, that we found a $\langle 5, 5, 5 \rangle$ TPP triple (S, T, U) in a group G of the C2 list. We only know that $\underline{R}(G) < 125$, 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 $\mathbf{A} \mapsto e_{\mathbf{A}}$ and $\mathbf{B} \mapsto e_{\mathbf{B}}$ of the matrices $\mathbf{A} = [a_{s,t}]$ and $\mathbf{B} = [b_{t,u}]$ in $\mathbb{C}[G]$:

$$a_{s,t} \mapsto a_{s,t} s^{-1} t, \quad b_{t,u} \mapsto b_{t,u} t^{-1} u \quad \text{for all } s \in S, t \in T, u \in U. \quad (5)$$

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

$$\mathbb{C}[G] \cong \mathbb{C}^{d_1 \times d_1} \times \mathbb{C}^{d_2 \times d_2} \times \dots \times \mathbb{C}^{d_\ell \times d_\ell}, \quad (6)$$

where d_1, \dots, d_ℓ are the character degrees of G . The given matrices \mathbf{A} and \mathbf{B} are now represented by ℓ -tuples of matrices $e_{\mathbf{A}} \mapsto (\mathbf{A}_1, \dots, \mathbf{A}_\ell)$ and $e_{\mathbf{B}} \mapsto (\mathbf{B}_1, \dots, \mathbf{B}_\ell)$. The last step is easy: just use the best known algorithms to compute the products $\mathbf{A}_i \mathbf{B}_i$ or *try to make use of the structures (e.g., symmetries, zero entries, ...) in \mathbf{A}_i and \mathbf{B}_i to find even better algorithms for the small products $\mathbf{A}_i \mathbf{B}_i$* . 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 \mathbf{A}_i and \mathbf{B}_i : There is space for d_i^2 entries in each small matrix. Over all small matrices together we have enough space for $d_1^2 + \dots + d_\ell^2 = |G|$ 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 (6), 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 (6) 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 $\min_e M(e)$?

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

$$\mathbb{C}[A_5] \cong \mathbb{C} \times \mathbb{C}^{3 \times 3} \times \mathbb{C}^{3 \times 3} \times \mathbb{C}^{4 \times 4} \times \mathbb{C}^{5 \times 5}.$$

We know that A_5 realizes $\langle 5, 5, 5 \rangle$. There is place for 60 elements in the embedding $e_A \in \mathbb{C}[A_5]$ of a 5×5 matrix A with 25 elements. The same for e_B . 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 (6) *completely*. Thus, only $49 - 1^2 - 3^2 - 3^2 - 4^2 = 14$ elements of the small matrices in $\mathbb{C}^{5 \times 5}$ 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 $A_5 B_5$ to beat 125 scalar multiplications.

Example. The symmetric group $G := S_3$ on three objects realizes $\langle 2, 2, 2 \rangle$ via the TPP triple $S = \{s_1 = 1_G, s_2 = (1, 2)\}$, $T = \{t_1 = 1_G, t_2 = (1, 3)\}$, $U = \{u_1 = 1_G, u_2 = (2, 3)\}$. We identify A_{ij} with A_{s_i, t_j} and B_{jk} with B_{t_j, u_k} . The transformation into $\mathbb{C}[G]$ results in

$$\begin{aligned} c_1 &:= a_{11}1_G + a_{12}(1, 3) + a_{21}(1, 2) + a_{22}(1, 3, 2), \\ c_2 &:= b_{11}1_G + b_{12}(2, 3) + b_{21}(1, 3) + b_{22}(1, 3, 2). \end{aligned}$$

The character degree pattern of S_3 is $(1^2, 2^1)$, so $\mathbb{C}[G] \cong \mathbb{C} \times \mathbb{C} \times \mathbb{C}^{2 \times 2}$. To construct the map $f: \mathbb{C}[G] \rightarrow \mathbb{C} \times \mathbb{C} \times \mathbb{C}^{2 \times 2}$, we follow [1, Example 13.37]. The irreducible representations of S_3 are

- (1) The trivial representation $\tau: S_3 \rightarrow \mathbb{C}, g \mapsto 1$.
- (2) The alternating representation $\alpha: S_3 \rightarrow \mathbb{C}, g \mapsto \text{sgn}(g)$.
- (3) The representation $\rho: S_3 \rightarrow \mathbb{C}^{2 \times 2}, (2, 3) \mapsto \begin{bmatrix} 1 & -1 \\ 0 & -1 \end{bmatrix}, (1, 2, 3) \mapsto \begin{bmatrix} 0 & -1 \\ 1 & -1 \end{bmatrix}$.

Thus, we conclude

$$f\left(\sum_{g \in S_3} \lambda_g g\right) = \left(\sum_{g \in S_3} \lambda_g \tau(g), \sum_{g \in S_3} \lambda_g \alpha(g), \sum_{g \in S_3} \lambda_g \rho(g)\right).$$

It follows that

$$\begin{aligned} f(c_1) &= \left(a_{11} + a_{12} + a_{21} + a_{22}, a_{11} + a_{22} - a_{12} - a_{21}, \begin{bmatrix} a_{11} - a_{22} - a_{12} & a_{21} + a_{22} \\ a_{21} - a_{22} - a_{12} & a_{11} + a_{12} \end{bmatrix} \right), \\ f(c_2) &= \left(b_{11} + b_{12} + b_{21} + b_{22}, b_{11} + b_{22} - b_{12} - b_{21}, \begin{bmatrix} b_{11} + b_{12} - b_{21} - b_{22} & b_{22} - b_{12} \\ -b_{21} - b_{22} & b_{11} - b_{12} + b_{21} \end{bmatrix} \right). \end{aligned}$$

In \mathbb{C} we compute the product with 1 multiplication. In $\mathbb{C}^{2 \times 2}$ we can use Strassen’s algorithm with 7 multiplications. Therefore, we need 9 multiplications to calculate $f(c_1)f(c_2)$.

This method provides a way to construct the multiplication algorithm induced by a given TPP triple. If it works (that means if one can answers questions (Q1), (Q2) and (Q3)), we find *new best* or at least *nontrivial* matrix multiplication algorithms for matrices of small dimension. Another approach to multiply matrices with a given TPP triple can be found in Gonzalez-Sanchez et al. [7]. But as far as we know, this approach doesn’t construct the matrix multiplication algorithm itself.

6. CONCLUSIONS

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 $\langle 5, 5, 5 \rangle$ 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 $\langle 6, 6, 6 \rangle$ 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 1.12?

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 $\langle 40, 40, 40 \rangle$. The group $G = C_n^3 \wr C_2$ realizes $\langle 2n(n-1), 2n(n-1), 2n(n-1) \rangle$ with the rank $R(G) = 2|G| - T(G) = 4n^6 - \frac{1}{2}(n^6 + 3n^3) = \frac{1}{2}n^3(7n^3 - 3)$, see [2, Section 2] for details. Thus, for $n = 5$ it realizes 40×40 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 $R(40) = R(2^3 \cdot 5) \leq R(2)^3 R(5) \leq 7^3 \cdot 100 = 34,300$ 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 438–449, 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-Èric Drevet, Md. Nazrul Islam, and Èric 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 Caravantes, 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 $\langle 5, 5, 5 \rangle$ 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 3×3 matrices using 23 multiplications. *Bull. Amer. Math. Soc.*, 82(1):126–128, 1976.
- [11] O. M. Makarov. A non-commutative algorithm for multiplying 5×5 matrices using one hundred multiplications. *U.S.S.R. Comput. Maths. 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 2×2 Matrices. *Linear Algebra and Appl.*, 4:381–388, 1971.