Factoring groups efficiently.

Neeraj Kayal[1] and Timur Nezhmetdinov[2]

[1] Microsoft Research Lab India,
196/36 2nd Main, Sadashivanagar Bangalore 560080 India
[2] Lehigh University, Pennsylvania, USA *

Abstract. We give a polynomial time algorithm that computes a decomposition of a finite group G given in the form of its multiplication table. That is, given G, the algorithm outputs two subgroups A and B of G such that G is the direct product of A and B, if such a decomposition exists.

1 Introduction

1.1 Background

Groups are basic mathematical structures and a number of computer algebra systems do computations involving finite groups. One very successful area of research has been the design of algorithms that handle a given permutation group G. It is customary to specify G via a small set of generating permutations. Despite the succinctnesness of such representations, a substantial polynomial-time machinery has developed for computing with permutation groups [Luk]. A major stimulus for this activity was the application to the graph isomorphism problem (it is easily seen for example that the graph isomorphism problem reduces to the problem of computing the intersection of two permutation groups). Ensuing studies resulted in algorithms for deciphering the basic building blocks of the group making available constructive versions of standard theoretical tools [BKL79], [Luk87], [KT88], [Kan85a], [Kan85b], [Kan90], [BSL87]. But problems such as subgroup intersection and computing sylow subgroups are trivial for verbosely encoded groups - groups which are specified via their multiplication tables. On the other hand, the verbose representation begets its own set of problems. The central problem here is to design a polynomial-time algorithm that given two finite groups G_1 and G_2 decides whether they are isomorphic or not. Towards this end, the classification theorem for finite simple groups helps us by giving a polynomial time algorithm for isomorphism of finite simple groups. What prevents us in going from simple groups to arbitary groups is our lack of understanding of 'group products' - how to put together two groups to get a new one. So far, despite much effort [Mil78,AT04,Gal09] not much progress has been made in resolving the complexity of the general group isomorphism problem. For

* This work was done while the second author was visiting DIMACS under REU program.

example, even for groups whose derived series has length two, we do not know how to do isomorphism testing in NP ∩ coNP. A less ambitious goal (a cowardly alternative?) is to develop algorithms that unravel the structure of a verbosely given group. In this work we devise an algorithm that accomplishes one such unravelling.

1.2 Direct product of groups

Given two groups A and B one of the most natural ways to form a new group is the direct product, denoted $A \times B$. As a set, the direct product group is the Cartesian product of A and B consisting of ordered pairs (a, b) and the group operation is component-wise.

$$(a_1, b_1) \cdot (a_2, b_2) = (a_1 \cdot a_2, b_1 \cdot b_2).$$

Given (the Cayley representation of) groups A and B, its trivial to compute (Cayley representation of) the group $G = A \times B$. In this article, we consider the inverse problem of factoring or decomposing a group G as a direct product of two of its subgroups. There are some very natural motivations for such a study. The fundamental theorem of finite abelian groups (Theorem 2) states that any finite abelian group can be written uniquely upto permutation as the direct product of cyclic groups of prime power order. This theorem means that the problem of finding an isomorphism between two given abelian groups [Kav07] is essentially the same as the problem of factoring an abelian group. In the general case, the Remak-Krull-Shcmidt theorem (Theorem 3) tells us that the factorization of a group as a direct product of indecomposable groups is "unique" in the sense that the isomorphism class of each of the components of the factorization is uniquely determined. This means that all such decompositions are structurally the same. This motivates us to devise an efficient algorithm which finds such a factorization. Computing the *direct factorization* of a group has also been found to be useful in practice. The computer algebra system GAP (Groups, Algorithms and Programming) implements an inefficient algorithm for this purpose [Hul].

1.3 Algorithm outline.

Notice that if we have an algorithm that efficiently computes *any nontrivial* factorization $G = A \times B$, we can also efficiently compute the complete factorization of G into indecomposable subgroups by recursing on A and B. Therefore we formulate our problem as follows: given a group G, find subgroups A and B such that $G = A \times B$ and both A and B are nontrivial subgroups of G. The algorithm is developed in stages, at each stage we solve a progressively harder version of the GroupDecomposition problem until we arrive at a complete solution to the problem. Each stage uses the solution of the previous stage as a subroutine.

- *G is abelian.* The proof of the fundamental theorem of finite abelian groups (Theorem 2) is constructive and gives a polynomial-time algorithm. This case has also been studied previously and a linear time algorithm is given in [CF07].

– *The subgroup A is known.* In this case we have to just find a B such that $G = A \times B$. We call it the GroupDivision problem and in section 4, an algorithm is devised in two substages.
  - B is abelian.
  - B is nonabelian.
– *A is unknown but an abelian A exists.* We call this the SemiAbelianGroupDecomposition problem and the algorithm is given in section 5.
– *A is unknown and all indecomposable direct factors of G are nonabelian.* In section 6 we describe the algorithm for this most general form of GroupDecomposition .

Let us now assume that we have an efficient algorithm for both GroupDivision and for SemiAbelianGroupDecomposition and outline the algorithm for solving GroupDecomposition using these subroutines. One of the main sources of difficulty in devising an efficient algorithm is that the decomposition of a group is not unique. Indeed, there can be superpolynomially many different decompositions of G. We fix a reference decomposition. We first analyze the different ways a group can decompose and come up with some invariants which do not depend on the particular decomposition at hand. Assume G is decomposable and let us fix a decomposition of G,

$$G = G_1 \times G_2 \times \ldots \times G_t.$$

with each G_i indecomposable. Let $Z_1 \stackrel{\text{def}}{=} \text{Cent}_G(G_2 \times \ldots \times G_t)$, where for any $A \subseteq G$, $\text{Cent}_G(A)$ denotes the subgroup of G consisting of all the elements of G that commutes with every element of A (it is called the centralizer of A in G). Our algorithm first computes Z_1 (the group Z_1 is invariant across different decompositions) and then uses this subgroup in order to solve GroupDecomposition . Notice that $Z_1 = G_1 \times \text{Cent}(G_2 \times \ldots \times G_t)$. By a repeated application of the subroutine SemiAbelianGroupDecomposition , we can obtain a decomposition of Z_1 into

$$Z_1 = H_1 \times Y,$$

where Y is an abelian group and H_1 has no abelian direct factors. An application of theorem 4 allows us to deduce that any such decomposition of Z_1 has the following properties:

1. H_1 is indecomposable and isomorphic to G_1.
2. $\exists Y_1 \trianglelefteq G$ such that $G = H_1 \times Y_1$.

Having obtained H_1, we obtain an appropriate Y_1 by invoking GroupDivision on (G, H_1) and thereby get a decomposition of G. We will now introduce some notation and then outline the procedure used to compute Z_1.

Notation. For a positive integer s, $[s]$ denotes the set $\{1, 2, \ldots, s\}$. We will denote the center of the group G by Z. For a set A of elements of G, $\langle A \rangle$ will denote the subgroup of G generated by the elements in A.

Computing Z_1. From the given group G, we construct a graph Γ_G which has the following properties:

1. The nodes of G correspond to conjugacy classes of G; however not all conjugacy classes of G are nodes of Γ_G. For a connected component Λ of Γ_G, let $\mathrm{Elts}(\Lambda) \subseteq G$ denote the set of all $g \in G$ that are members of some conjugacy class occuring in Λ.
2. (Proposition 3). If a decomposition of G contains t nonabelian indecomposable components then the number of connected components in Γ_G is at least t.
3. (Proposition 3). Let $G = G_1 \times \ldots \times G_t$, with each G_i indecomposable. Let Z be the center of G and let $\Lambda_1, \ldots, \Lambda_s$ be the set of connected components of Γ_G. Then there exists a partition

$$[s] = S_1 \uplus \ldots \uplus S_t$$

such that for any i,

$$G_i \pmod{Z} = \prod_{j \in S_i} \langle \mathrm{Elts}(\Lambda_j) \rangle \pmod{Z}.$$

4. (Proposition 5). The number of connected components $s \leq \log |G|$ and G/Z has a decomposition given by

$$G/Z = \langle \mathrm{Elts}(\Lambda_1) \rangle \pmod{Z} \times \ldots \times \langle \mathrm{Elts}(\Lambda_s) \rangle \pmod{Z}.$$

Now given only the group G and the constructed graph Γ_G, we do not the set $S_1 \subseteq [s]$ apriori. But $s \leq \log |G|$, so we can simply iterate over all possible sets S_1 in just $|G|$ iterations. Let us therefore assume that we have the appropriate S_1. Then the sought-after set Z_1 can be obtained as follows:

$$Z_1 \stackrel{\mathrm{def}}{=} \mathrm{Cent}_G(\bigcup_{j \notin S_1} \mathrm{Elts}(\Lambda_j)).$$

This completes the outline of the algorithm. Let us summarize the algorithm.

Algorithm I. GroupDecomposition

Input. A group G in the form of a Cayley table.

1. Construct the conjugacy class graph Γ_G associated to the group G.
2. Compute the connected components $\Lambda_1, \ldots, \Lambda_s$ of Γ_G.
3. For each $S_1 \subseteq [s]$ do the following:
   (a) Let $Z_1 \stackrel{\mathrm{def}}{=} \mathrm{Cent}_G(\left\langle \bigcup_{j \notin S_1} \mathrm{Elts}(\Lambda_j) \right\rangle)$.
   (b) By repeated invocations to SemiAbelianGroupDecomposition determine $H_1, Y \trianglelefteq G$ such that $Z_1 = H_1 \times Y$ and H_1 has no abelian direct factors and Y is abelian.
   (c) Invoke GroupDivision on (G, H_1) to determine if there exists a $Y_1 \trianglelefteq G$ such that $G = H_1 \times Y_1$. If such a Y_1 is found then **output** (H_1, Y_1).
4. If no decomposition has been found, **output** *NO SUCH DECOMPOSITION.*

2 Preliminaries.

2.1 Notation and Terminology.

Cent(G) will denote the center of a group G and $|G|$ its size. For an element $a \in G$, we will denote $|\langle a \rangle|$ by ord(a). We will denote the conjugacy class of the element a by $\mathcal{C}_a$, i.e.

$$\mathcal{C}_a \stackrel{\text{def}}{=} \{g \cdot a \cdot g^{-1} \mid g \in G\} \subset G.$$

Let $A, B \subseteq G$. We write $A \leq G$ when A is a subgroup of G and $A \trianglelefteq G$ when A is a normal subgroup of G. $\text{Cent}_G(A)$ will denote the subgroup of elements of G that commute with every element of A. i.e.

$$\text{Cent}_G(A) \stackrel{\text{def}}{=} \{g \in G \mid a \cdot g = g \cdot a \ \ \forall a \in A\}.$$

We will denote by $[A, B]$ the subgroup of G generated by the set of elements

$$\{a \cdot b \cdot a^{-1} \cdot b^{-1} | a \in A, \ b \in B\}.$$

We shall denote by $A \cdot B$ the set

$$\{a \cdot b \mid a \in A, b \in B\} \subseteq G.$$

We say that a group G is *decomposable* if there exist nontrivial subgroups A and B such that $G = A \times B$ and *indecomposable* otherwise. When A is a normal subroup of G we will denote by $B \pmod A$ the set of cosets $\{A \cdot b \mid b \in B\}$ of the quotient group G/A. We will say that a subgroup A of G is *a direct factor* of G if there exists another subgroup B of G such that $G = A \times B$ and we will call B a *direct complement* of A.

The canonical projection endomorphisms. When a group G has a decomposition

$$G = G_1 \times G_2 \times \ldots \times G_t$$

then associated with this decomposition is a set of endomorphisms $\pi_1, \ldots, \pi_t$ of G with

$$\pi_i : G \mapsto G_i, \quad \pi_i(g_1 \cdot g_2 \cdot \ldots \cdot g_t) = g_i.$$

where $g = g_1 \cdot g_2 \cdot \ldots \cdot g_t \in G$ $(\forall i \in [t] : \ g_i \in G_i)$ is an arbitary element of G. The π_i's we call the *canonical projection endomorphisms* of the above decomposition.

2.2 Background.

Theorem 1. *(Expressing G as a direct product of A and B, cf. [Her75]) Let G be a finite group and A, B be subgroups of G. Then $G = A \times B$ if and only if the following three conditions hold:*

- *Both A and B are normal subgroups of G.*
- $|G| = |A| \cdot |B|$.

– $A \bigcap B = \{e\}$.

Theorem 2. *(The fundamental theorem of finite abelian groups, cf. [Her75]) Every finite abelian group G can be as the written product of cyclic groups of prime power order.*

Theorem 3. *(Remak-Krull-Schmidt, cf. [Hun74]) Let G be a finite group. If*

$$G = G_1 \times G_2 \times \ldots \times G_s$$

and

$$G = H_1 \times H_2 \times \ldots \times H_t$$

with each G_i, H_j indecomposable, then $s = t$ and after reindexing $G_i \cong H_i$ for every i and for each $r < t$,

$$G = G_1 \times \ldots \times G_r \times H_{r+1} \times \ldots \times H_t.$$

Notice that the uniqueness statement is stronger than simply saying that the indecomposable factors are determined upto isomorphism.

3 Invariants of group factorization.

The main source of difficulty in devising an efficient algorithm for the decomposition of a group lies in the fact that the decomposition need not be unique. Let us therefore analyze what one decomposition should be in reference of another.

Lemma 1. *For a group G, suppose that $G = A \times B$. Then for a subset $C \subset G$,*

$G = C \times B \iff C = \{\alpha \cdot \phi(\alpha) \mid \alpha \in A\}$, *where $\phi : A \mapsto Cent(B)$ is a homomorphism.*

Theorem 4. *(Characterization of the various decompositions of a group.) Let G be a finite group with*

$$G = G_1 \times G_2 \times \ldots \times G_t \tag{1}$$

with each G_i indecomposable. For $i \in [t]$, define M_i to be the normal subgroup of G as follows:

$$M_i \stackrel{\text{def}}{=} G_1 \times \ldots \times G_{i-1} \times G_{i+1} \times \ldots \times G_t,$$

so that $G = G_i \times M_i \forall i \in [t]$. If G has another decomposition

$$G = H_1 \times H_2 \times \ldots \times H_t \tag{2}$$

(the number of H_j's must equal t by Theorem 3) with each H_j indecomposable, then there exist t homomorphisms $\{\phi_r : G_r \mapsto Cent(M_r)\}_{r \in [t]}$ so that after reindexing, for each $r \in [t]$,

$$H_r = \{\alpha \cdot \phi_r(\alpha) \mid \alpha \in G_r,\ \phi_r(\alpha) \in Cent(M_r)\}$$

4 An algorithm for GroupDivision

In this section we solve the group division problem which is used in step 3 of Algorithm I. Let us recall that GroupDivision is the following problem: given a group G and a normal subgroup $A \trianglelefteq G$, find a $B \trianglelefteq G$ such that $G = A \times B$, if such a decomposition exists. We will solve this problem itself in two stages. First, we devise an efficient algorithm assuming that the quotient group G/A is abelian and then use this as a subroutine in the algorithm for the general case.

4.1 When the quotient group G/A is abelian.

In this case we can assume that $G = A \times B$ where B is abelian. Observe that in this case, for every coset $A \cdot g$ of A in G, we can pick an element $b \in A \cdot g$ such that $b \in \mathrm{Cent}(G)$ and $\mathrm{ord}_G(b) = \mathrm{ord}_{G/A}(A \cdot g)$. Also, the quotient group G/A is abelian and therefore using the abelian group decomposition algorithm, we can efficiently find a complete decomposition of G/A. So let

$$G/A = \langle A \cdot g_1 \rangle \times \ldots \times \langle A \cdot g_t \rangle$$

Now from each coset $A \cdot g_i$ we pick a representative element b_i such that $b_i \in \mathrm{Cent}(G)$ and $\mathrm{ord}_G(b_i) = \mathrm{ord}_{G/A}(A \cdot b_i)$. For any such set of b_i's, its an easy verification that $G = A \times \langle b_1 \rangle \times \ldots \times \langle b_t \rangle$.

4.2 When the quotient group G/A is nonabelian.

We first give the algorithm and then prove its correctness.

Algorithm II. GroupDivision

Input. A group G and a normal subgroup A of G.

Output. A subgroup C of G such that $G = A \times C$, if such a C exists.

1. Compute $T \stackrel{\text{def}}{=} \left\langle \{a \cdot g \cdot a^{-1} \cdot g^{-1} \mid a \in \mathrm{Cent}_G(A), g \in G\} \right\rangle$.
2. If T is not a normal subgroup of G then **output** *NO SUCH DECOMPOSITION.*
3. Compute $\tilde{G} \stackrel{\text{def}}{=} G/T$ and $\tilde{A} \stackrel{\text{def}}{=} \{T \cdot a \mid a \in A\} \trianglelefteq \tilde{G}$.
4. Verify that $T \bigcap A = \{e\}$. If not, output *NO SUCH DECOMPOSITION.* If yes, then we deduce that the canonical map $a \mapsto Ta$ is an isomorphism from A to $\tilde{A}$.
5. Using the abelian group division algorithm given above, determine if there exists a $\tilde{B} \trianglelefteq \tilde{G}$, *with* $\tilde{B}$ *abelian*, so that $\tilde{G} = \tilde{A} \times \tilde{B}$. If so, determine elements $Tg_1, Tg_2, \ldots, Tg_t \in G/T$ such that
$$\tilde{G} = \tilde{A} \times \langle Tg_1 \rangle \times \langle Tg_2 \rangle \times \ldots \times \langle Tg_t \rangle .$$
6. From each coset Tg_i, pick *any* representative element c_i. Compute $C \stackrel{\text{def}}{=} \langle T \bigcup \{c_1, \ldots, c_t\} \rangle \leq G$.
7. If $G = A \times C$ then **output** C else **output** *NO SUCH DECOMPOSITION.*

The algorithm clearly has polynomial running time and it remains for us to prove its correctness. To see whats going on in the algorithm above, let us assume that $G = A \times B$ and fix this decomposition of G. Its easy to verify that the subgroup T computed in step 1 is a normal subgroup of G and $T = [B, B]$. Also, $T = [B, B] \subseteq B$ and therefore $A \bigcap T$ must be $\{e\}$. This implies that the canonical mapping $a \mapsto T \cdot a$ is an isomorphism from A to $\tilde{A}$ This explains step 4 of the algorithm. Observe that the $\tilde{G}$ computed in step 3 has the decomposition

$$\tilde{G} = \tilde{A} \times (B/[B, B]).$$

But $B/[B, B]$ is an abelian group so we can use the previous algorithm and decompose $\tilde{G}$ into product of $\tilde{A}$ times a number of cyclic groups. By the end of step 6, we would have computed $c_1, \ldots, c_t \in G$ such that

$$\tilde{G} = \tilde{A} \times \tilde{C}, \quad \text{where } \tilde{C} \stackrel{\text{def}}{=} \langle Tc_1 \rangle \times \langle Tc_2 \rangle \times \ldots \times \langle Tc_t \rangle \leq G.$$

Proposition 1. *$C \trianglelefteq G$ and the elements of C and A together generate G. Furthermore, $C \bigcap A = \{e\}$.*

Summarizing, we have A and C are normal subgroups of G that span G and have a trivial intersection which means that $G = A \times C$, as required to prove the correctness of the algorithm.

5 An algorithm for SemiAbelianGroupDecomposition

In this section, we solve the special case of GroupDecomposition when some of the indecomposable components of G are abelian groups. That is given G, we wish to find an abelian subgroup B and another subgroup A of G so that

$$G = A \times B, \quad \text{where } B \text{ is abelian.} \tag{3}$$

Since B is abelian, it has a decomposition into a direct product of cyclic groups. So let

$$B = \langle b_1 \rangle \times \ldots \times \langle b_t \rangle .$$

so that G becomes

$$G = A \times \langle b_1 \rangle \times \ldots \times \langle b_t \rangle .$$

Thus, if G has a decomposition of the form (3) then there exists a $b \in G$ such that $\langle b \rangle$ is a direct factor of G. Conversely, to find a decomposition of the form (3) it is sufficient to find a b such that $\langle b \rangle$ is a direct factor of G. Knowing B, we can find an appropriate direct complement of $\langle b \rangle$ efficiently using the algorithm for GroupDivision given previously. Lastly, given the group G, we find an appropriate b in polynomial-time by iterating over all the elements of G and using the algorithm for GroupDivision to verify whether $\langle b \rangle$ is a direct factor of G or not.

6 The conjugacy class graph of a group and its properties.

Here we give the construction of the *conjugacy class graph* of a group. Consider a group G which has a decomposition

$$G = A \times B.$$

Fixing this decomposition, consider the conjugacy class $\mathcal{C}_g$ of an arbitary element $g = \alpha \cdot \beta \in G$, where $\alpha \in A, \beta \in B$. Observe that $\mathcal{C}_g = \mathcal{C}_\alpha \cdot \mathcal{C}_\beta$ and the elements of $\mathcal{C}_\alpha$ and $\mathcal{C}_\beta$ commute. More generally, we have

Observation 5. *If $G = G_1 \times \ldots \times G_t$ and $g = g_1 \cdot \ldots \cdot g_t$ is an arbitary element of G, with each $g_i \in G_i$ then*

$$\mathcal{C}_g = \mathcal{C}_{g_1} \cdot \mathcal{C}_{g_2} \cdot \ldots \cdot \mathcal{C}_{g_t}.$$

Furthermore for all $i \neq j$ each element of $\mathcal{C}_{g_i}$ commutes with every element of $\mathcal{C}_{g_j}$.

For the rest of this section, we fix the group G and a reference decomposition

$$G = G_1 \times G_2 \times \ldots \times G_t.$$

Let $\{\pi_i : G \mapsto G_i \mid i \in [t]\}$ be the set of canonical projection endomorphisms associated with the above decomposition. If any of the G_i's are abelian groups then we can obtain a decomposition of G using the algorithm for SemiAbelianGroupDecomposition given in section 5. So henceforth we will assume that all the G_i's are nonabelian. Observation 5 above motivates the following definitions.

Definition 1. *We say that two conjugacy classes $\mathcal{C}_a$ and $\mathcal{C}_b$* commute *when for every $\alpha \in \mathcal{C}_a$ and $\beta \in \mathcal{C}_b$, α and β commute.*

Definition 2. *Call a conjugacy class* reducible *$\mathcal{C}_g$ if it is either a conjugacy class of an element from the center of G, or there exist two conjugacy classes $\mathcal{C}_a$ and $\mathcal{C}_b$ such that*

- *Neither a nor b belongs to the center of G.*
- *$\mathcal{C}_a$ and $\mathcal{C}_b$ commute.*
- *$\mathcal{C}_g = \mathcal{C}_a \cdot \mathcal{C}_b$*
- *$|\mathcal{C}_g| = |\mathcal{C}_a| \cdot |\mathcal{C}_b|$*

If a conjugacy class is not reducible, then call it irreducible.

Proposition 2. *If a conjugacy class $\mathcal{C}_g$ is irreducible then there exists a unique $i \in [t]$ such that $\pi_i(g) \notin Cent(G_i)$.*

Proof. If it happens that for all $i \in [t]$, $\pi_i(g) \in \mathrm{Cent}(G_i)$ then $g \in \mathrm{Cent}(G)$ so that the conjugacy class $\mathcal{C}_g$ is reducible by definition. If more than one $\pi_i(G)$ are noncentral elements then by observation 5, we would get that $\mathcal{C}_g$ is reducible.

The converse of this proposition is not true in general. The above proposition implies that corresponding to a conjugacy class $\mathcal{C}_g$, there exists a unique G_i such that $\pi_i(g) \notin \mathrm{Cent}(G_i)$. Let us call this subgroup G_i the *indecomposable component associated to* the conjugacy class $\mathcal{C}_g$. Let us now define the *conjugacy class graph* Γ_G associated to a group G.

Definition 3. *The* graph *of a group G (denoted Γ_G) is a graph with irreducible conjugacy classes as nodes and such that a pair of nodes is connected by an edge iff the corresponding pair of conjugacy classes does* not *commute.*

The connected components of Γ_g can be computed efficiently and they give us information about the direct factors of G.

Proposition 3. *Let $\Lambda_1, \ldots, \Lambda_s$ be the connected components of Γ_G. Then $s \geq t$ and there is a partition*

$$[s] = S_1 \uplus S_2 \uplus \ldots \uplus S_t$$

such that $\left\langle \cup_{i \in S_j} Elts(\Lambda_i) \right\rangle \pmod Z = G_j \pmod Z$ for all j where $Z = Cent(G)$.

Proof. Let us consider two irreducible conjugacy classes $\mathcal{C}_g$ and $\mathcal{C}_h$. Let the indecomposable components associated with $\mathcal{C}_g$ and $\mathcal{C}_h$ be G_i and G_j respectively. Suppose that $i \neq j$. Then $\pi_j(g)$ and $\pi_i(h)$ are central elements of G so that every element of $\mathcal{C}_g$ commutes with every other element of $\mathcal{C}_h$. Thus there is no edge between the nodes corresponding to $\mathcal{C}_g$ and $\mathcal{C}_h$. This implies that if $\mathcal{C}_g$ and $\mathcal{C}_h$ are in the same connected component of Γ_G then the indecomposable components associated with $\mathcal{C}_g$ and $\mathcal{C}_h$ are the same. Each nonabelian component of G gives rise to at least one irreducible conjugacy class so that the number of connected components s of Γ_G is at least the number of indecomposable nonabelian components of G. The above argument shows that there exists a partition of $[s]$ into S_i such that $\left\langle \cup_{i \in S_j} \Lambda_i \right\rangle \pmod Z \subseteq G_j \pmod Z$ for all j. The inclusion is in fact an equality because all the irreducible conjugacy classes generate all the noncentral elements of G by construction.

In general it is not true that the number of connected components of Γ_G equals the number of indecomposable nonabelian groups in the factorization of G. The irreducible conjugacy classes of each of the G_i may be divided into more than one component. However we have the following:

Proposition 4. *If the center of group G is trivial, then the number of connected components of its graph is equal to the number of indecomposable groups in the factorization of G. Moreover, the subgroups generated by the conjugacy classes of each of the components are normal disjoint subgroups which together span G; thus we have the factorization of G.*

Using the proposition we can efficiently factor a group with a trivial center. When the center of the group is non-trivial it is no longer the case that each component of Γ_G generates one of the factors in the factorization of G. We would need to search through the partitions of the set of connected components to find the components associated to an indecomposable factor, say G_1. For that we need a bound on the number of components of the graph:

Proposition 5. *The number of connected components is upper bounded by* $\log |G|$.

7 Putting everything together

We now have all the component steps of Algorithm I. So let us conclude with the proof of correctness of Algoritm I.

Theorem 6. *If the input to Algorithm I is a decomposable group G then it necessarily computes a nontrivial decomposition of G, otherwise it outputs* NO SUCH DECOMPOSITION*. Moreover, Algorithm I has running time polynomial in $|G|$.*

Proof. Clearly, if the group is indecomposable our algorithm outputs *NO SUCH DECOMPOSITION*. By Proposition 5, $s \leq \log |G|$ so that the number of iterations in step 3 is at most $|G|$. All the operations inside the loop (steps 3a to 3c) are polynomial-time computable so that the overall running time also poly($|G|$). It remains to show that if G is decomposable then our algorithm outputs a nontrivial decomposition. Let the given group G have a decomposition

$$G = G_1 \times \ldots \times G_t \tag{4}$$

with each G_i indecomposable. Let Z be the center of G. In the algorithm we iterate over all subsets of the connected components of Γ_G so let us assume that we have found the subset S_1 of indices of connected components corresponding to conjugacy class of elements of G_1. By Proposition 5 we have

$$G_2 \times G_3 \times \ldots \times G_t \pmod{Z} = \left\langle \cup_{j \notin S_1} \mathrm{Elts}(\Lambda_j) \right\rangle \pmod{Z}.$$

This means that in step 3a we would have computed

$$\begin{aligned} Z_1 &\stackrel{\text{def}}{=} \mathrm{Cent}_G(\langle \cup_{j \notin S_1} \mathrm{Elts}(\Lambda_j) \rangle) \\ &= \mathrm{Cent}_G(G_2 \times G_3 \times \ldots \times G_t) \\ &= G_1 \times \mathrm{Cent}(G_2) \times \mathrm{Cent}(G_3) \times \ldots \times \mathrm{Cent}(G_t) \end{aligned}$$

Let us now consider the decomposition $Z_1 = H_1 \times Y$ obtained in step 3b of Algorithm I. By the Remak-Krull-Schmidt theorem (Theorem 3), all decompositions of Z_1 are isomorphic so that if H_1 is any direct factor of Z_1 which has no abelian direct factors then H_1 must be indecomposable and isomorphic to G_1. Furthermore by an application of theorem 4, we must have that H_1 must be of the form

$$H_1 = \{\alpha \cdot \phi(\alpha) \mid \alpha \in G_1,\ \phi(\alpha) \in (\mathrm{Cent}(G_2) \times \mathrm{Cent}(G_3) \times \ldots \times \mathrm{Cent}(G_t))\},$$

where $\phi : H_1 \mapsto \mathrm{Cent}(G_2) \times \mathrm{Cent}(G_3) \times \ldots \times \mathrm{Cent}(G_t)$ is a homomorphism. By lemma 1, we can replace G_1 by H_1 in the factorization (4) so that in fact

$$G = H_1 \times G_2 \times G_3 \times \ldots \times G_t.$$

In particular, this means that H_1 is a direct factor of G so that in step (3c), using the algorithm for GroupDivision , we necessarily recover a nontrivial factorization of G.

References

[AT04] Arvind and Toran. Solvable group isomorphism is (almost) in NP intersect coNP. In *Proceedings of the 19th Annual Conference on Computational Complexity*, pages 91–103, 2004.

[BKL79] L. Babai, W. M. Kantor, and E. M. Luks. Computational complexity and the classification of finite simple groups. In *Proceedings of the 24th FOCS*, pages 162–171, 1979.

[BSL87] L. Babai, A. Seress, and E. M. Luks. Permutation groups in nc. In *Proceedings of 19th STOC*, pages 409–420, 1987.

[CF07] Li Chen and Bin Fu. Linear and sublinear time algorithms for the basis of abelian groups. In *Electronic Colloquium on Computational Complexity, Technical report TR07-052*, 2007.

[Gal09] Francois Le Gall. Efficient isomorphism testing for a class of group extensions. In *Proceedings of the annual Symposium on Theoretical Aspects of Computer Science*, 2009.

[Her75] I. N. Herstein. *Topics in Algebra*. John Wiley & Sons, New York, 2nd edition, 1975.

[Hul] Alexander Hulpke. Gap project repository. available at `http://www.math.colostate.edu/~hulpke/gapproj/projects.htm`.

[Hun74] Thomas W. Hungerford. *Algebra*. Number 73 in Graduate Texts in Mathematics. Springer-Verlag, New York, 1974.

[Kan85a] W. M. Kantor. Polynomial-time algorithms for finding elements of prime order and sylow subgroups. *Journal of Algorithms*, 6:478–514, 1985.

[Kan85b] W. M. Kantor. Sylow's theorem in polynomial time. *J. Comp. Syst. Sci.*, 30:359–394, 1985.

[Kan90] W. M. Kantor. Finding sylow normalizers in polynomial time. *Journal of Algorithms*, 11:523–563, 1990.

[Kav07] T. Kavitha. Linear time algorithms for abelian group isomorphism and related problems. *J. Comput. Syst. Sci.*, 73(6):986–996, 2007.

[KT88] W. M. Kantor and D. E. Taylor. Polynomial-time versions of sylow's theorem. *Journal of Algorithms*, 9:1–17, 1988.

[Luk] Eugene Luks. Lectures on polynomial-time computation in groups. available at `http://ix.cs.uoregon.edu/~luks`.

[Luk87] Eugene M. Luks. Computing the composition factors of a permutation group in polynomial time. *Combinatorica*, 7:87–99, 1987.

[Mil78] Gary Miller. On the nlog n isomorphism technique. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, 1978.