Nearly Linear Time Isomorphism Algorithms for Some Nonabelian Group Classes

Bireswar Das(✉) and Shivdutt Sharma(✉)

IIT Gandhinagar, Gandhinagar, India
{bireswar,shiv.sharma}@iitgn.ac.in

Abstract. The isomorphism problem for groups, when the groups are given by their Cayley tables is a well-studied problem. This problem has been studied for various restricted classes of groups. Kavitha gave a linear time isomorphism algorithm for abelian groups (JCSS 2007). Although there are isomorphism algorithms for certain nonabelian group classes, the complexities of those algorithms are usually super-linear. In this paper, we design linear and nearly linear time isomorphism algorithms for some nonabelian groups. More precisely,

- We design a linear time algorithm to factor Hamiltonian groups. This allows us to obtain an $\mathcal{O}(n)$ algorithm for the isomorphism problem of Hamiltonian groups, where n is the order of the groups.
- We design a nearly linear time algorithm to find a maximal abelian factor of an input group. As a byproduct we obtain an $\tilde{\mathcal{O}}(n)$ isomorphism for groups that can be decomposed as a direct product of a nonabelian group of bounded order and an abelian group, where n is the order of the groups.

1 Introduction

Two groups $(G, \cdot)$ and $(H, \times)$ are said to be isomorphic if there exists a bijective function $f : G \longrightarrow H$, which is a homomorphism i.e. $\forall a, b \in G, f(a \cdot b) = f(a) \times f(b)$. The decision version of this problem is to check whether two input groups $(G, \cdot)$ and $(H, \times)$ are isomorphic or not. There are multiple ways in which a group can be given as the input. Two of commonly used methods are by a generating set and by the Cayley table. The complexity of the group isomorphism problem varies with the input representation. In this paper we assume that input groups are given by their Cayley tables unless stated otherwise explicitly.

It is not known whether the group isomorphism problem (GrISO) is in P. If it is NP-complete then polynomial hierarchy collapses at the second level [4]. Tarjan (see e.g., [14]) gave an $n^{\log n+\mathcal{O}(1)}$ algorithm for GrISO. While this still remains the best upper bound for the general group isomorphism problem, progress has been made for restricted classes of groups. For solvable groups, Arvind and Torán showed that the problem is in $\mathsf{NP} \cap \mathsf{co\text{-}NP}$ under a reasonable complexity

© Springer Nature Switzerland AG 2019
R. van Bevern and G. Kucherov (Eds.): CSR 2019, LNCS 11532, pp. 80–92, 2019.
https://doi.org/10.1007/978-3-030-19955-5_8

theoretic assumption [1]. Rosenbaum and Wagner gave a $n^{1/2(\log n)+\mathcal{O}(1)}$ algorithm for the isomorphism problem of p-groups [17] and Rosenbaum gave an $n^{1/2(\log n)+\mathcal{O}(\log n/\log\log n)}$ time algorithm for solvable groups [16].

The isomorphism problem for various restricted classes of groups has been studied in the past [1,3,8,9,15,19,20]. Efficient polynomial time algorithms for the isomorphism problem of abelian groups were designed by Savag [18], and Vikas [20]. Kavitha gave a remarkable linear time algorithm for the isomorphism problem of abelian groups [19]. Kavitha's paper also provides us with a useful tool for computing the orders of all the elements in *any* group in linear time.

Polynomial time algorithms have been designed for some classes of nonabelian groups. For example, Le Gall designed an efficient algorithm for groups consisting of a semidirect product of an abelian group with a cyclic group of coprime order [7]. Later, Qiao, Sarma, and Tang gave a polynomial time algorithm for the isomorphism problem of groups with normal Hall subgroups [15]. Babai, Codenotti and Qiao gave a polynomial time algorithm for the class of groups with no normal abelian subgroups [2]. We believe that motivation behind these results on the isomorphism problem of nonabelian groups was to enlarge the family of group classes for which polynomial time algorithm is known. To the best of our knowledge the runtime of these algorithms are superquadratic.

Our goal in this paper is to design linear or nearly linear time algorithm for some nontrivial classes of nonabelian groups.

The isomorphism problem of nilpotent class 2 groups[1] is *not* known to be in polynomial time. A nonabelian group is Hamiltonian if all of its subgroups are normal. These groups are nilpotent class 2 groups. We design an $\mathcal{O}(n)$ algorithm for the recognition and the isomorphism problem of the Hamiltonian groups where n is the size of the input groups.

A Hamiltonian group is a direct product of the quaternion group Q_8 and an abelian group with certain structure [5]. This motivates us to study the class of groups that can be decomposed as a direct product of any arbitrary nonabelian group of bounded order and an abelian group without any specific structure. We design an $\tilde{\mathcal{O}}(n)$ algorithm for the recognition and the isomorphism problem of such groups.

Kayal and Nezhmetdinov gave an algorithm to factorize an input group into a direct product of indecomposable factors [12]. We note that this group factorization algorithm combined with any of the polynomial time isomorphism algorithms for abelian group gives us a polynomial time isomorphism test for the group classes considered in this paper. However, direct application of the result by Kayal and Nezhmetdinov only gives us superquadratic isomorphism algorithms. One of the contributions of this paper is to use the structure of the input groups to tweak and bypass some of the computation heavy steps of the algorithm by Kayal et al. [12].

The main results of this paper are stated below.

Theorem 1. *There exists an $\mathcal{O}(n)$ algorithm for the recognition and the isomorphism problem of Hamiltonian groups where n is the size of the input groups.*

[1] A group G is *nilpotent class 2* if $G/Z(G)$ is abelian.

Theorem 2. *There exists an $\tilde{O}(n)$ algorithm for the recognition and the isomorphism problem of groups that can be decomposed as a direct product of a nonabelian group of bounded order and an abelian group where n is the size of the input groups.*

2 Preliminaries

In this section, we describe some of the group-theoretic definitions and background used in the paper. For more details see [6,10,13,19]. For a group G, the number of elements in G or the *order* of G is denoted by $|G|$. Let $x \in G$ be an element of group G, then $\mathrm{ord}_G(x)$ denotes the order of the element x in G, which is the smallest power i of x such that $x^i = e$, where e is the identity element of the group G.

For a subset $S \subseteq G$, $\langle S \rangle$ denotes the subgroup generated by the set S. For a subgroup $A \leq G$, *the centralizer* of A, denoted $C_G(A)$, is the set $\{g \in G \mid ag = ga, \forall a \in A\}$. The *center* $Z(G)$ of group G is the subgroup with elements $\{g \in G \mid ga = ag, \forall a \in G\}$.

Given $H \leq G$, the *normal closure* of H in G is the smallest normal subgroup of G containing H and is denoted by $\langle H \rangle^G$. The *commutator* subgroup $[G, G]$ of a group G is the subgroup $\langle \{xyx^{-1}y^{-1} \mid \forall x, y \in G\} \rangle$.

Let G be a finite group and A, B be subgroups of G. Then G is a *direct product* of A and B, denoted $G = A \times B$, if (1) $A \trianglelefteq G$ and $B \trianglelefteq G$, (2) $|G| = |A||B|$, (3) $A \cap B = \{e\}$. 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. We say that a subgroup A of G is a *direct factor* (or *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* (or *complement*) of A.

The fundamental theorem for finitely generated abelian groups implies that a finite group G can be decomposed as a direct product $G = G_1 \times G_2 \times \ldots \times G_t$, where each G_i is a cyclic group of order p^j for some prime p and integer $j \geq 1$. If a_i generates the cyclic group G_i for $i = 1, 2, 3, \ldots, t$ then the elements $a_1, a_2, \ldots, a_t$ are called a *basis* of G. An elementary abelian p-group is an abelian group in which every nontrivial element has order p. Chen and Fu [6], and Karagiorgos and Poulakis [11] gave linear time algorithms for finding a basis of abelian groups.

Theorem 3 (Remak-Krull-Schmidt, see e.g., [10]). *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 any $r < t$, $G = G_1 \times \ldots \times G_r \times H_{r+1} \times \ldots \times H_t$.*

A *Remak-Krull-Schmidt decomposition* of a group G is a decomposition such that each direct factor of group G is indecomposable. The following lemma establishes a relationship between two different decompositions of a group G in which one of the factors is same.

Lemma 1 **([12]).** *For a group G, suppose that $G = H \times K$. Then for a $K' \trianglelefteq G$, $G = H \times K'$ if and only if $K' = \{\alpha\phi(\alpha) \mid \alpha \in K\}$, where $\phi : K \longrightarrow Z(H)$ is a homomorphism.*

Model of Computation: Our model of computation is same as that of many of the algorithms for groups given by Cayley table (e.g., [6,19]). It is a RAM model where random access can be done in constant time. Each register and memory unit can store $\mathcal{O}(\log |G|)$ bits. The arithmetic, logic and comparison operations on $\mathcal{O}(\log |G|)$ bits take constant time. Unless stated otherwise we assume that the elements of the group are encoded as $1, 2, \ldots, |G|$.

Nearly Linear Time Algorithms: A group theoretic algorithm is *nearly linear time* if it has runtime $\mathcal{O}(|G| \log^{\mathcal{O}(1)} |G|)$. We hide the logarithmic factor by using the notation $\tilde{\mathcal{O}}(|G|)$. We list some useful nearly linear time algorithms for group theoretic problems for groups given by their Cayley tables in the next lemma. The ideas behind these results are either known as folklores or directly follows from easy observations.

Lemma 2. *1. Given $S \subseteq G$, one can compute the elements of the subgroup $\langle S \rangle$ in $\mathcal{O}(|G| \log |G|)$ time.*
2. Finding an $\mathcal{O}(\log |G|)$ sized generating set can be done in $\mathcal{O}(|G| \log |G|)$ time.
3. For a group G the center $Z(G)$ can be computed in $\mathcal{O}(|G| \log |G|)$ time.
4. Given $A \leq G$, one can check whether $A \trianglelefteq G$ in $\mathcal{O}(|G| \log |G|)$ time.
5. Given two subgroups A and B of G, one can check whether $G = A \times B$ in $\mathcal{O}(|G| \log |G|)$ time.

Proof. We present a sketch of the proof for the statements *1* and *2*. The proof for the other statements are easy.

Consider a directed graph $X = (V, E)$, where $V = G$ and $E = \{(g, gs) | g \in G, s \in S\}$. Let $H = \langle S \rangle$. Finding H amounts to computing the set R of vertices reachable from the identity element $e \in G$. Notice that R is also a strongly connected component with $|H||S|$ edges. Thus, the runtime for computing the strongly connected component is $\mathcal{O}(|H||S|)$. This proves (*1*) if $|S| \leq \log |G|$. We use similar ideas as in the proof of *2* to handle the case when $|S| > \log |G|$. Hence, we first prove (*2*).

For *2*, we pick an element $a \in G \setminus \{e\}$ and set $S_1 = \{a\}$. The algorithm keeps on computing sets $S_1, S_2, \ldots$ in stages as follows. At the ith stage we have the set S_i. If we discover $G = \langle S_i \rangle$ we stop the algorithm and output S_i. Otherwise we pick $g \in G \setminus \langle S_i \rangle$ and let $S_{i+1} = S_i \cup \{g\}$. Note that $2|\langle S_i \rangle| \leq |\langle S_{i+1} \rangle|$ as $\langle S_{i+1} \rangle$ contains the disjoint cosets $\langle S_i \rangle$ and $\langle S_i \rangle g$. Thus, if the last set is S_r, then $r \leq \log |G|$. Let $\langle S_i \rangle = H_i$ and $n_i = |H_i|$. Computing H_i via finding a suitable strongly connected component in a graph as mentioned above takes time $\mathcal{O}(|H_i||S_i|) = \mathcal{O}(i|H_i|)$. Furthermore, we note that $n_r = |G|$ and $n_i \leq n_r/2^{r-i}$. Thus, computing all the H_is together takes time $\mathcal{O}(\sum_i in_i)$ which is $\mathcal{O}(|G| \log |G|)$. Finding an element $g \in G \setminus \langle S_i \rangle$ takes time $\mathcal{O}(|G|)$ and this too happens at most $\log |G|$ times. Thus, the runtime of the algorithm is $\mathcal{O}(|G| \log |G|)$.

To complete the proof of *1*, we modify the algorithm for finding a logarithmic sized generating set by setting $S_1 = \{a\}$ for any $a \in S$ and then picking the new elements g from $S \setminus \langle S_i \rangle$ instead of $G \setminus \langle S_i \rangle$.

Organization of the Paper: In Sect. 3 we give algorithms to compute quotient groups in linear time. In Sect. 4 we discuss some nearly linear time algorithms for finding complements of certain groups. This results are then used in Sects. 5 and 6.

3 Algorithms in Quotient Groups

Suppose we have the list of elements of a group G and a black-box for the group multiplication. Let N be a normal subgroup of G (also given as a list or array). In this section we show how to construct the quotient structure G/N in linear time. More precisely, we describe a linear time algorithm to build a data structure that can serve as a black-box to compute multiplications in G/N. Once we have the data structure, a multiplication query in G/N can be processed in constant time with just one query to the black-box for G.

Many of the algorithms for groups given by their Cayley table work in the same running time if the algorithm has access to the list of group elements and a group multiplication black-box. As a consequence of this quotient black-box construction we can see that the algorithms by Kavitha [19], Chen and Fu [6], and Karagiorgos and Poulakis [11] still run in linear time in quotient groups.

Suppose the list of elements of a group G and a normal subgroup N of G are given along with a black-box for G. We construct lists L_i for $i = 1, \ldots, |G/N|$, each containing the elements of different cosets of N in G in Algorithm 1. We also compute the minimum elements m_i (in the input order) of the list L_i for each i.

Algorithm 1. Computing lists corresponding to the cosets of N in G.

```
1  **Input** : A group $G$ and a normal subgroup $N$ of $G$;
2  **Find** : Lists $L_i$'s and the elements $m_i$'s as described above;

3  Create an array *flag* indexed by the elements of $G$ and set
     $flag[g] = 0, \forall g \in G$;
4  $i \leftarrow 0$;
5  **for** $g \in G$ **do**
6      **if** $flag[g] = 0$ **then**
7          $i \leftarrow i + 1$;
8          Prepare a list $L_i$ of size $|G/N|$ with the elements of $Ng$;
9          $m_i \leftarrow min\ L_i$;
10         **for** $g_1 \in Ng$ **do**
11             $flag[g_1] = 1$;
12         **end**
13     **end**
14 **end**
```

Run-Time Analysis of Algorithm 1: In the algorithm *flag*$[g]$ = *1* line 6 indicates that we have already processed the coset containing g and no further action is required. If $flag[g] = 0$ then the algorithm spends $\mathcal{O}(|G/N|)$ time within the "if" condition. But in the process it also discovers all the elements of the coset Ng for which the "if" condition will *not* be executed in future. This shows that the run-time of Algorithm 1 is $\mathcal{O}(|G|)$. □

It is easy to see that in linear time we can compute an array S of size G and indexed by the elements of G such that $S[g] = i$, where L_i is the list produced by by Algorithm 1 containing the elements of Ng.

The data structure for the quotient group G/N consists of the lists L_i's, the sequence $m_1, m_2, \dots, m_{|G/N|}$ and the array S along with an access to the black-box for G. The elements of G/N will be, as usual, $1, 2, \dots, |G/N|$. The element i corresponds to the list L_i, which in turn corresponds to one of the cosets of N in G. If we need to compute the product of i and j, we first compute $m = m_i * m_j$ using the multiplication black-box for G and then return $S[m]$. By construction $S[m]$ is the index of the list containing the coset elements of the coset Nm.

We notice that any bounded number of repeated quotient construction can be done in linear time using the above method.

4 Algorithms for Finding Complements

Given a group G and a normal subgroup D of G, a complement of D in G is a subgroup B such that $G = D \times B$. It is important to note that a complement of a subgroup may or may not exist, and even if a complement exists it may not be unique. Kayal and Nezhmetdinov [12] gave an algorithm for finding a complement of a given normal subgroup D of G. Their algorithm is divided into two cases: G/D is abelian and G/D is nonabelian. The result for the first case can be stated as follows.

Theorem 4 ([12]). *There is an algorithm to check if a complement of a normal subgroup D of a group G exists in $\tilde{\mathcal{O}}(|G|)$ time, when G/D is abelian. The algorithm also returns a complement if it exists.*

Proof Sketch: A careful analysis of the complement finding algorithm given in [12] using the results from [6,19] shows that it takes $\tilde{\mathcal{O}}(|G|)$ time to find a complement of the subgroup D in G (if it exists). We use the linear time quotient construction techniques from Sect. 3 multiple times. Additionally, we use the facts that computing $Z(G)$ and testing normality can be done in $\tilde{\mathcal{O}}(|G|)$ time by Lemma 2.

In the second case when G/D is nonabelian, it is not clear how to make the algorithm by Kayal and Nezhmetdinov [12] run in nearly linear time in general (see [12]). Fortunately, for the purpose of this paper, as we would see in Sect. 6, we only need to deal with the subcase when D is a subgroup of the center $Z(G)$ of G. During its execution the algorithm in [12] computes a quotient group which can be done in linear time using results in Sect. 3.

The algorithm by Kayal and Nezhmetdinov computes a group $T = \langle\{aga^{-1}g^{-1} \mid a \in C_G(D), g \in G\}\rangle$. One can verify that except for this, all other steps in the algorithm can be made to run in $\tilde{\mathcal{O}}(|G|)$ time without the assumption $D \leq Z(G)$. It is the computation of T where we use a different approach using the fact that $D \leq Z(G)$ to obtain the desired nearly linear runtime. We first mention an easy observation.

Observarion 1. *If $D \leq Z(G)$ then $C_G(D) = G$.*

From Observation 1, it is immediate that $T = \langle\{aga^{-1}g^{-1} \mid a \in G, g \in G\}\rangle$ which is nothing else but the commutator subgroup $[G, G]$ of G. Lemma 3 gives us a way to compute T efficiently.

Lemma 3 (see e.g., [13]). *If $G = \langle S\rangle$ then $[G, G] = \langle[S, S]\rangle^G$, where S is a generating set of G and $[S, S] = \{aga^{-1}g^{-1} \mid a, g \in S\}$.*

We can compute a generating set S of size $\mathcal{O}(\log|G|)$ of G in time $\mathcal{O}(|G|\log|G|)$ by Lemma 2. Again by Lemma 2 we can compute an $\mathcal{O}(\log|G|)$ sized generating set for the group $\langle[S, S]\rangle$ in $\mathcal{O}(|G|\log|G|)$ time. Let us denote this set by T_{gen}. Algorithm 2 given below computes a generating set for $[G, G]$ (see [13]).

Algorithm 2. Algorithm to find an $\mathcal{O}(\log|G|)$ sized generating set of $[G, G]$

```
1 Input : A group $G = \langle S\rangle$ and $T_{gen}$ (defined above);
2 Find : An $\mathcal{O}(\log|G|)$ sized generating set of $[G, G]$;

3 Let $K \leftarrow \langle T_{gen}\rangle$;
4 while $\exists b \in T_{gen}, a \in S$ such that $a^{-1}ba \notin K$ do
5         $T_{gen} \leftarrow T_{gen} \cup \{a^{-1}ba\}$ and $K \leftarrow \langle T_{gen}\rangle$;
6 return $T_{gen}$
```

Runtime Analysis of Algorithm 2: Each time a new generator is added to T_{gen} the size of the group $K = \langle T_{gen}\rangle$ is at least doubled, which implies that the number of iterations of the while loop is $\mathcal{O}(\log|G|)$. We maintain the group K as an array A_K indexed by the group elements $g \in G$ such that $A_K[g] = 1$ if and only if $g \in K$. Thus, for any $a \in S$ and $b \in T_{gen}$, we can check if $a^{-1}ba \notin K$ in $\mathcal{O}(1)$ time. It takes $\mathcal{O}(|G|\log|G|)$ time to compute the group $\langle T_{gen}\rangle$. Now it is easy to verify that the overall runtime of Algorithm 2 is $\mathcal{O}(|G|(\log|G|)^3)$. $\square$

It is important to note that the inverse of an element $a \in G$ can be found in $\mathcal{O}(|G|)$ time (step 4). However since the number of iterations is only $\mathcal{O}(\log|G|)$, we would need to compute the inverse of $\mathcal{O}(\log|G|)$ many elements, which implies that the overall runtime to find inverses is $\mathcal{O}(|G|\log|G|)$.

Summarising the above discussion we obtain:

Theorem 5. *There exists an algorithm to find a complement of a subgroup D of the center $Z(G)$ of a groups G in time $\tilde{\mathcal{O}}(|G|)$ whenever a complement exists.*

5 Hamiltonian Group Recognition and Isomorphism

A Hamiltonian group is a nonabelian group all of whose subgroups are normal. Since every subgroup of such a group is normal, it follows that there is a unique Sylow subgroup of any fixed order. In this section we consider the following problem.

HAMILTONIAN GROUP RECOGNITION
Input : Given a group $(G, \cdot)$ by its Cayley table.
Find : Is G a Hamiltonian group?

The following structure theorem is one of the main ingredients for our result.

Theorem 6 ([5, page 114]). *Let G be a Hamiltonian group. Then*

- *G is the quaternion group Q_8; or,*
- *G is the direct product of Q_8 and B, or of Q_8 and A, or of Q_8, B and A, where A is an abelian group of odd order and B is an elementary 2-group. Moreover, every such direct product is a Hamiltonian group.*

We recall that the quaternion group Q_8 is a nonabelian group with eight elements and it is generated by two elements a and b with the conditions $a^4 = 1, a^2 = (ab)^2 = b^2$ (see e.g., [5]). An elementary 2-group is isomorphic to $\mathbb{Z}_2^k$ for some k. Thus, from Theorem 6 we can see that the Sylow 2-subgroup of a Hamiltonian group is $Q_8 \times \mathbb{Z}_2^k$ for some nonnegative integer k and the other Sylow subgroups are all abelian. The theorem also implies that Hamiltonian groups are nilpotent. The Sylow decomposition can be computed in $\mathcal{O}(|G|)$ time using methods[2] described in [6]. Next we decompose the Sylow 2-subgroup using Algorithm 3. If we find that the Sylow 2-subgroup is not of the form $Q_8 \times \mathbb{Z}_2^k$ for some k, then we can immediately conclude that the input group is not Hamiltonian. Otherwise, we can use the techniques developed by Kavitha [19] to test if the odd order Sylow subgroups are abelian. If that is the case then we know that the input group is Hamiltonian. Moreover, since the odd order Sylow subgroups are abelian, the algorithm given by Chen and Fu in [6] or Karagiorgos and Poulakis in [11] also give us a Remak-Krull-Schmidt decomposition of the odd order Sylow subgroups. The decomposition of the Sylow 2-subgroup obtained from Algorithm 3 along with the decomposition of the odd order abelian Sylow subgroups gives us a Remak-Krull-Schmidt decomposition of the input Hamiltonian group.

Given a Sylow 2-subgroup as input, Algorithm 3 checks if it is Hamiltonian and also returns a Remak-Krull-Schmidt decomposition isomorphic to $Q_8 \times \mathbb{Z}_2^k$ if the input is indeed Hamiltonian. We use the next lemma in the algorithm.

[2] We can compute the Sylow decomposition in $\mathcal{O}(|G|)$ *without* using the result given [6], if G is Hamiltonian 2-group. Note that in a Hamiltonian 2-group order of each non-trivial element will be either 2 or 4.

Lemma 4. *Any two non-commutating elements in a Hamiltonian 2-group generate a quaternion group which is also a direct factor.*

Proof. Let G be a Hamiltonian 2-group and let $g, g' \in G$ be two non-commutating elements. To show that $\langle g, g' \rangle \cong Q_8$ it is enough to show that $g^4 = 1, g^2 = (gg')^2 = g'^2$. As G is a Hamiltonian 2-group, $G = Q_8 \times \mathbb{Z}_2^k$ for some k. Thus, we can write $g = (a_1, b_1)$ and $g' = (a_2, b_2)$, where $a_1, a_2 \in Q_8$ and $b_1, b_2 \in \mathbb{Z}_2^k$. It is easy to verify that $g^4 = 1, g^2 = (gg')^2 = g'^2$. Now we prove that $\langle g, g' \rangle$ is also a factor of G. Let $C = \{\alpha\phi(\alpha) \mid \alpha \in A\}$, where $\phi : Q_8 \longrightarrow \mathbb{Z}_2^k$ is a homomorphism that maps the generators a_1 and a_2 of Q_8 to b_1 and b_2 respectively. Now using Lemma 1, we can see that $G = C \times \mathbb{Z}_2^k$. Moreover, it is an easy verification to see that $C = \langle g, g' \rangle$. $\square$

Algorithm 3. Algorithm for the recognition and decomposition of Hamiltonian 2-groups

```
1  Input : A group $(G, \cdot)$;
2  Decide : Is $G$ a Hamiltonian 2-group?

3  if $G \cong Q_8$ then stop and return Hamiltonian 2-group, and $G$ ;
4  $P = \{g \in G \mid ord_G(g) = 4\}$;
5  if $P = \emptyset$ then report "Not Hamiltonian 2-group";
6  Pick any $g \in P$;
7  Find an element $g' \in P$ such that $gg' \neq g'g$. If no such pair exists then
     report "Not Hamiltonian 2-group";
8  if $\langle g, g' \rangle \cong Q_8$ then
9      Compute a complement $C$ of $\langle g, g' \rangle$ in $G$;
10     if $C$ exists and it is an elementary abelian 2-group then
11         return $C$;
12     end
13 end
14 else
15     report "Not Hamiltonian 2-group";
16 end
```

We now prove the correctness of the algorithm and give the run-time analysis. Checking whether $G \cong Q_8$ or not can be done in $\mathcal{O}(1)$ time. From now on, we assume that $G \ncong Q_8$. In a Hamiltonian 2-group, all non-central elements are of order 4 and constitutes the set P. Since we are interested in elements of order 4, P can be computed in linear time even without using results in [19].

Since the picked element $g \in P$ (Line 5) is non-central, there must exist an element $g' \in P$ such that $gg' \neq g'g$. If no such pair is found in P then G is not a Hamiltonian 2-group. Otherwise by Lemma 4, $\langle g, g' \rangle \cong Q_8$ and will also be direct factor of G. Thus, if the check $\langle g, g' \rangle \cong Q_8$ fails we conclude that G is not a Hamiltonian 2-group.

Using Kavitha's result given in [19], we can test whether C is abelian in time $\mathcal{O}(|G|)$ (Line 9). If C is abelian and all the elements of C have order 2, then we conclude that C is an elementary abelian 2-group and the algorithm returns the complement C.

Finally we argue that we can also compute a complement of $\langle g, g' \rangle$ in time $\mathcal{O}(|G|)$ in Line 8. We can use the result of Theorem 4 to find a complement. However, a direct application of Theorem 4 would only give us an $\tilde{\mathcal{O}}(|G|)$ upper bound. Below we show that the structure of Hamiltonian group could be used to get an $\mathcal{O}(|G|)$ upper bound.

The major time consuming computation tasks inside the complement finding algorithm of Theorem 4 are computing the quotient group $G/\langle g, g' \rangle$, computing the center and testing normality (see [12]).

Since $\langle g, g' \rangle$ is the quaternian group of order 8, testing its normality in time $\mathcal{O}(|G|)$ is trivial. We can compute $G/\langle g, g' \rangle$ in $\mathcal{O}(|G|)$ time using techniques discussed in Sect. 3. If G is a Hamiltonian 2-group, then $G/\langle g, g' \rangle$ will be an abelian group. The task of checking whether $G/\langle g, g' \rangle$ is abelian can be performed in $\mathcal{O}(|G|)$ time using the algorithm described in [19]. If G is a Hamiltonian 2-group, then the center of the group G consists of all order 2 elements along with the identity. One can find all these elements in $\mathcal{O}(|G|)$ time. If the original group is not a Hamiltonian 2-group then the final test, which is to confirm if we have actually computed a valid decomposition (see Section 4.1 of [12], last line) will identify any error that might have occurred in the computation of the center. The final test to verify the validity of the computed decomposition can be performed in linear time exploiting the structure of Hamiltonian 2-groups. These observations imply that Theorem 4 can be modified to find a complement of $\langle g, g' \rangle$ in G in $\mathcal{O}(|G|)$ time.

Once we have the Remak-Krull-Schmidt decompositions of two Hamiltonian groups the isomorphism test is trivial.

Theorem 7. *There exists an algorithm that given two Hamiltonian groups G and H tests if they are isomorphic in time $\mathcal{O}(|G|)$.*

6 Groups with a Bounded Nonabelian Direct Factor

Taking motivation from Hamiltonian groups, which are direct product of the nonabelian quaternion group Q_8 and an abelian group, we study the recognition and the isomorphism problem of a more general class of groups which can be decomposed as a direct product of a nonabelian group of bounded order and an abelian group. For a fixed d, let $\mathcal{G}_d = \{G \mid G = A \times B, \text{ where } |A| \leq d \text{ and } B \text{ is abelian}\}$. It is easy to see that the isomorphism problem for groups in $\mathcal{G}_d$ can be solved in linear time once we have a decomposition of each of the input groups as a direct product of a small nonabelian group with no cyclic factor and an abelian group.

In this section we show that given a nonabelian group G, it can be decomposed as a direct factor of a nonabelian group with *no cyclic factor* and an abelian group in nearly linear time. We note that for this algorithm we *do not need any upper bound* on the size of the nonabelian factor. The idea is to keep on peeling off direct cyclic factors from the given group as long as possible. Each time we factor out a cyclic group, the size of the other factor decreases by at least half. Thus, the process of factoring out cyclic groups can happen for at most $\log |G|$ iterations. Next we define the CYCLIC FACTOR problem below.

CYCLIC FACTOR
Input : A group $(G, \cdot)$ given by its Cayley table.
Find : A cyclic factor $\langle b \rangle$ and $H \trianglelefteq G$ (if they exist) such that $G = \langle b \rangle \times H$.

We show that the CYCLIC FACTOR problem can be solved in $\tilde{\mathcal{O}}(|G|)$ time. From this result and the above discussion we can immediately obtain the following theorem.

Theorem 8. *There is an algorithm that takes the Cayley table of a nonabelian group G as input and in time $\tilde{O}(|G|)$ returns two groups A and B, such that $G = A \times B$ where A is a nonabelian group with no cyclic factor and B is abelian.*

In the rest of the section we focus on the CYCLIC FACTOR problem. The following lemma helps us to solve the problem.

Lemma 5. *If G has a cyclic factor then for any basis $\{b_1, b_2, \ldots, b_\ell\}$ of $Z(G)$, there is $i \in [\ell]$ such that $\langle b_i \rangle$ is a factor of G.*

Proof. Let $G = A \times B$, where A is nonabelian with no cyclic factor and B is abelian. Notice that $Z(G) = Z(A) \times B$. Let $Z(A) = \langle c_1 \rangle \times \ldots \times \langle c_r \rangle$ and $B = \langle d_1 \rangle \times \ldots \times \langle d_k \rangle$ be a basis decomposition of $Z(A)$ and B. This gives a basis decomposition $Z(G) = \langle c_1 \rangle \times \ldots \times \langle c_r \rangle \times \langle d_1 \rangle \times \ldots \times \langle d_k \rangle$. Let $c'_1, \ldots, c'_r, b'_1, \ldots, b'_k$ be an another basis of $Z(G)$, where c'_is and b'_js are ordered to satisfy the following conditions from Remak-Krull-Schmidt theorem:

(i) $\langle c'_i \rangle \cong \langle c_i \rangle, \forall i \in [r]$ and $\langle b'_j \rangle \cong \langle d_j \rangle, \forall j \in [k]$, and

(ii) $Z(G) = \langle c_1 \rangle \times \ldots \times \langle c_r \rangle \times B_p = Z(A) \times B_p$,

where $B_p = \langle b'_1 \rangle \times \ldots \times \langle b'_k \rangle$. By Lemma 1, we have $B_p = \{\alpha\phi(\alpha) \mid \alpha \in B\}$ for some homomorphism $\phi : B \longrightarrow Z(Z(A)) = Z(A)$. The same lemma can be used once more to show that $G = A \times B_p$. The result follows if we take $\{c'_1, \ldots, c'_r, b'_1, \ldots, b'_k\} = \{b_1, b_2, \ldots, b_\ell\}$. □

The above lemma immediately suggests an algorithm to solve the CYCLIC FACTOR problem for a given group G: (i) Find the center $Z(G)$ of the group G, (ii) Compute a basis $\{b_1, b_2, \ldots, b_\ell\}$ of $Z(G)$, and (iii) Try to find a complement of $\langle b_i \rangle$ in G for all $i = 1, 2, \ldots, \ell$.

In step (iii) of the algorithm if we find a complement then we have solved the problem. On the other hand, if G has a cyclic factor then Lemma 5 ensures that the algorithm would find a cyclic factor. This shows the correctness of the algorithm.

We now discuss the runtime of the algorithm. We can use Kavitha's result to check if G is abelian in linear time [19]. If G is abelian, then cyclic factors of G can be found in $\mathcal{O}(|G|)$ time using results from [6]. Let us assume that input group G is nonabelian.

We can compute the center of a group in nearly linear time using Lemma 2. Thus, step (i) of the algorithm takes $O(|G| \log |G|)$ time.

Since $Z(G)$ is abelian we can use the linear time algorithm of Chen and Fu [6] or Karagiorgos and Poulakis [11] for step (ii) of the algorithm. Notice that the number of basis elements of $Z(G)$ is at most $\log |G|$. Thus, the maximum number of iterations in step (iii) is at most $\log |G|$. In general, we do not know how to compute a complement of a subgroup in nearly linear time. However, the fact that each $\langle b_i \rangle$ is a subgroup of the center of the group allows us to use Theorem 5. Thus, the runtime of step (iii) as well as the whole algorithm is $\tilde{O}(|G|)$.

References

1. Arvind, V., Torán, J.: Solvable group isomorphism is (almost) in NP $\cap$ coNP. ACM Trans. Comput. Theory **2**(2), 4:1–4:22 (2011)
2. Babai, L., Codenotti, P., Qiao, Y.: Polynomial-time isomorphism test for groups with no abelian normal subgroups. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012. LNCS, vol. 7391, pp. 51–62. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31594-7_5
3. Babai, L., Qiao, Y.: Polynomial-time isomorphism test for groups with abelian sylow towers. In: STACS 2012 (29th Symposium on Theoretical Aspects of Computer Science), vol. 14, pp. 453–464. LIPIcs (2012)
4. Boppana, R.B., Hastad, J., Zachos, S.: Does co-NP have short interactive proofs? Inf. Process. Lett. **25**(2), 127–132 (1987)
5. Carmichael, R.D.: Introduction to the Theory of Groups of Finite Order. GINN and Company, Boston (1937)
6. Chen, L., Fu, B.: Linear and sublinear time algorithms for the basis of abelian groups. Theor. Comput. Sci. **412**(32), 4110–4122 (2011)
7. Gall, F.L.: Efficient isomorphism testing for a class of group extensions. In: 26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26–28, 2009, Freiburg, Germany, Proceedings, pp. 625–636 (2009)
8. Garzon, M., Zalcstein, Y.: On isomorphism testing of a class of 2-nilpotent groups. J. Comput. Syst. Sci. **42**(2), 237–248 (1991)
9. Grochow, J.A., Qiao, Y.: Algorithms for group isomorphism via group extensions and cohomology. SIAM J. Comput. **46**(4), 1153–1216 (2017)
10. Hungerford, T.W.: Algebra. Graduate Texts in Mathematics, vol. 73. Springer, New York (1974)
11. Karagiorgos, G., Poulakis, D.: Efficient algorithms for the basis of finite abelian groups. Discrete Math. Algorithms Appl. **3**(04), 537–552 (2011)
12. Kayal, N., Nezhmetdinov, T.: Factoring groups efficiently. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 585–596. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02927-1_49
13. Luks, E.M.: Lectures on polynomial-time computation in groups. University of Oregon, Department of Computer and Information Science (1990)
14. Miller, G.L.: On the $n^{\log n}$ isomorphism technique (a preliminary report). In: Proceedings of the Tenth Annual ACM symposium on Theory of computing, pp. 51–58. ACM (1978)
15. Qiao, Y.M., Sarma, J., Tang, B.S.: On isomorphism testing of groups with normal hall subgroups. J. Comput. Sci. Technol. **27**(4), 687–701 (2012)

16. Rosenbaum, D.J.: Breaking the $\mathcal{O}(n \log n)$ barrier for solvable-group isomorphism. In: Proceedings of The Twenty-fourth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1054–1073. Society for Industrial and Applied Mathematics (2013)
17. Rosenbaum, D.J., Wagner, F.: Beating the generator-enumeration bound for p-group isomorphism. Theor. Comput. Sci. **593**, 16–25 (2015)
18. Savage, C.D.: An $\mathcal{O}(n^2)$ algorithm for abelian group isomorphism. Computer Studies [Program], North Carolina State University (1980)
19. Kavitha, T.: Linear time algorithms for abelian group isomorphism and related problems. J. Comput. Syst. Sci. **73**(6), 986–996 (2007)
20. Vikas, N.: An $\mathcal{O}(n)$ algorithm for abelian p-group isomorphism and an $\mathcal{O}(n \log n)$ algorithm for abelian group isomorphism. J. Comput. Syst. Sci. **53**(1), 1–9 (1996)