Electronic Colloquium on Computational Complexity, Report No. 52 (2011)

On the Complexity of Group Isomorphism

Fabian Wagner
Institut für Theoretische Informatik,
Universität Ulm, 89069 Ulm, Germany
fabian.wagner@uni-ulm.de

April 4, 2011

Abstract

The group isomorphism problem consists in deciding whether two groups G and H given by their multiplication tables are isomorphic. An algorithm for group isomorphism attributed to Tarjan runs in time $n^{\log n + O(1)}$, c.f. [Mil78].

Miller and Monk showed in [Mil79] that group isomorphism can be many-one reduced to isomorphism testing for directed graphs. For groups with n elements, the graphs have valence at least n. We many-one reduce group isomorphism onto graph isomorphism, such that the valence of the graphs is at most $d+1$ if d is the size of the largest factor group in a composition series of the groups. Combining this with the fact that isomorphism testing for graphs of valence d can be solved in time $n^{O(d)}$, p-group isomorphism is in time n^{cp} for a constant c.

We extend this algorithm to work for general groups, improving the exponential runtime behavior if factor groups of large size exist. Then we also consider the following simple group isomorphism algorithm, namely we compute a composition series and then guess coset representatives as generators. This algorithm has the same worst-case behavior as Tarjans algorithm. But if we combine both algorithms then we can show that group isomorphism is in time $n^{c \log n / \log \log n}$ for a constant c.

1 Introduction

Two groups G, H with ground set $\Omega = \{1, \ldots, n\}$ are *isomorphic* if there is a mapping $\phi : \Omega \to \Omega$ such that $\phi(i) \cdot \phi(j) = \phi(i \circ j)$ where we assume that $\circ$ is the group operation in G and $\cdot$ in H. The *group isomorphism problem* is to decide whether two groups G and H are isomorphic. We consider finite groups given in *table representation*, i.e. a table of $n \times n$ entries where the entry (i, j) contains the product ij.

The complexity of the group isomorphism problem has been studied for more than three decades. When finite groups are given by their generating sets then graph isomorphism is polynomial time Turing reducible to the permutation group isomorphism problem which is known to be in $\mathsf{NP} \cap \mathsf{coAM}$ [Luk93].

In [Mil79], isomorphism testing of explicitly given structures is many-one reduced to directed graph isomorphism. Groups can also be represented in multiplication tables and equivalently as a ternary relation R, i.e. $(a, b, c) \in R$ iff $ab = c$. As a corollary, group isomorphism can be reduced to graph isomorphism. In this reduction, the graphs have unbounded valence.

The following algorithm for isomorphism testing on two groups G, H with n elements given in table representation runs in time $n^{\log n + O(1)}$: compute a generating set A of size $\log n$ for G. Then

ISSN 1433-8092

try all mappings from A bijectively onto each possible subset A' of H. There are $\binom{n}{|A|} \cdot |A|!$ many such ordered sets. With the multiplication table of G, check for each such map whether it extends to an isomorphism from G onto H. We start with A and A' as partial ordered sets and recursively check whether $\phi(i) \cdot \phi(j) = \phi(i \circ j)$ is consistent with A and A'. We extend the partial ordered sets whenever $\phi(i \circ j)$ is a new element. This algorithm is attributed to Tarjan, c.f. [Mil78] and it is improved by Lipton, Snyder and Zalcstein [LSZ76] as a sharper $O(\log^2 n)$ space algorithm.

Arvind and Torán [AT04] give a 2-round Arthur-Merlin protocol for the group non-isomorphism problem such that Arthur and Merlin use $O(\log^6 n)$ random bits and $O(\log^2 n)$ non-deterministic bits, respectively. They derandomize this protocol in the case of solvable groups.

For abelian groups when given as generating sets, isomorphism can be tested in linear time [Kav03], it is hard for ModL and contained in $\mathsf{ZPL}^{\mathsf{ModL}}$ [AV04]. When given in multiplication tables then isomorphism testing is trivially in L, and also in $\mathsf{TC}^0(\mathsf{FOLL})$ (see [CTW10]). On the one side, not far from abelian appear the hardest cases, namely nilpotent groups of class 2. For these groups, the center $Z(G)$ and the quotient $G/Z(G)$ are abelian. On the other side, Babai et al. [BCGQ11] consider groups without abelian normal subgroups. They prove that isomorphism testing for this class of groups is in time $n^{O(1)+c \log\log n}$. With parameter $t(G)$, defined to be the smallest t such that each minimal normal subgroup of G has at most t simple groups, isomorphism testing is in time $n^{O(1)+c\log(t(G))}$ for a constant c. Quiao, Sarma and Tang [QMT11] present a framework to test isomorphism of groups with at least one normal Hall subgroup, when groups are given as multiplication tables.

Our contribution. We combine the classical and a new algorithm to improve the complexity of group isomorphism breaking the $n^{\log n}$ barrier.

The new algorithm many-one reduces group isomorphism onto graph isomorphism. If the largest factor group in a composition series for the input groups has size d then the valence of the so obtained graphs is $d + 1$.

It is well known that a composition series can be obtained in polynomial time for a group G as follows: We compute the *socle* $Soc(G)$ of G, this is a normal subgroup isomorphic to a direct product of simple groups. The factor group $G/Soc(G)$ is then decomposed recursively. For groups with n elements, the composition series has length at most $\log n$. We describe this decomposition in Section 4 and use it for the following algorithms for group isomorphism.

Before, we start with some notions. Let $\mathsf{seq}(S)$ denote the sequence of composition factors given by a composition series S. In $\mathsf{seq}(S)$, the composition factors are partially ordered primarily according to the decomposition process of G and as second criterion according to the isomorphism types of the factor groups.

1. *Tarjans algorithm with known composition series.* Isomorphism testing for two groups G and H can be done in two steps. First, we compute a composition series S for G and S' for H such that if G is isomorphic to H then also $(G, \mathsf{seq}(S))$ is isomorphic to $(H, \mathsf{seq}(S'))$ respecting the structure, i.e. mapping the subgroups in S of G blockwise onto those in S' of H. Second, we fix coset representatives as generators for each of the factor groups in S'. Then we run through all possibilities to select coset representatives respectively for S. If the mapping of the generators induces an isomorphism then accept. Let the factor groups in S have size at most p. This algorithm requires $n^{\log_p n}$ time.

2. *Reduction to graph isomorphism.* The second algorithm also has two steps. The first step is as before, the second step is different: We many-one reduce group isomorphism to graph isomorphism with graphs of valence at most $p+1$. In the reduction, the graph is based on a complete tree where each node has p children. We have several copies of this tree and also of a further graph gadget to encode the group multiplications. Isomorphism testing for valence-d graphs is in time $n^{O(d)}$ [BL83]. Let the factor groups in S have size at most p. Hence, the group isomorphism algorithm runs then in time $n^{c(p+1)}$.

3. *Combination of both algorithms.* We take the first algorithm to find generators for the corresponding factor groups of size $> \alpha$ and we take the second algorithm for the remaining factor groups of size $\leq \alpha$. For a constant c, the algorithm runs in time $n^{\log_\alpha n} + n^{c(\alpha+1)}$.

The first algorithm runs faster if the composition length is small, i.e. when the factor groups have large size. The second algorithm is an improvement especially for groups with factor groups of small size, e.g. for p-groups where p is a small prime. The runtime of the third algorithm becomes minimal if we set $\alpha = \log n / \log\log n$. In Section 6 we prove the main theorem.

Theorem 1.1 *Group isomorphism for groups with n elements given in table representation is in time (for a constant c):*

$$n^{c \log n / \log\log n}$$

In Section 3 we explain the reduction. In Section 5, we introduce the algorithm and describe the reduction to bounded valence GI when p-groups are given. We prove that the graphs in the reduction are *cone graphs* of logarithmic depth. That is, every vertex has a unique simple path of length at most $O(\log n)$ to a designated root vertex. Note, that Millers reduction to graph isomorphism does not bound the valence of the graphs [Mil79]. We prove the following theorem.

Theorem 1.2 *p-group isomorphism is polynomial time many-one reducible to valence $p+1$ depth $O(\log n)$ cone graph isomorphism.*

Isomorphism testing for valence-d graphs is in time $n^{O(d)}$ [BL83], we get the following theorem.

Theorem 1.3 *Group isomorphism for p-groups with n elements given in table representation is in time (for a constant c):* n^{cp}

2 Preliminaries

Groups. A *group* $G = (\Omega, \circ)$ is a set Ω together with an operation $\circ$, i.e. a 2-ary function, which satisfy the axioms of closure and associativity, G has a unique identity element e, and unique inverse elements. We also write in short $g \in G$ and mean that $g \in \Omega$ and we write for a product $g \circ h$ in short gh. We consider finite groups where Ω consists of n elements, i.e. we say that G has *order* or *size* $|G| = n$.

For an integer i, g^i is the element $g \in \Omega$ multiplied i times with itself. If $g^i = e$ for the smallest $i \geq 1$, then i is the *order* of g in the group G, in short we also write $\mathsf{ord}(g) = i$. The set $\{g, g^2, \ldots, g^{i-1}\}$ is denoted the *powers* of g. The element g^{-1} is the *inverse* element of g, it satisfies the equation $g^{-1}g = e$.

We write $H \leq G$ to denote that H is a *subgroup* of G. We use the following notion $G \setminus H = \{g \in G \mid g \notin H\}$ which has a different meaning than that of factor groups G/H which we will define below.

Let $g \in G$, then $gH = \{gh \mid h \in H\}$ is a *left coset* of H in G, and $Hg = \{hg \mid h \in H\}$ is a *right coset* of H in G. Any pair of left cosets (resp. right cosets) have the property that they contain either exactly the same set of elements or are disjoint. We write G as the union of its cosets

$$G = H + g_2H + \cdots + g_rH$$

to indicate that the cosets $H, g_2H, \ldots, g_rH$ are disjoint and exhaust G. The elements $g_2, \ldots, g_r$ are the *coset representatives*, these are arbitrary elements from each coset. We write H in short for eH, and assume that the identity e is the representative for H. We use these notions also when given left cosets. A set of representatives of all the cosets is called a *transversal*.

A group given in *table representation*, also denoted *Cayley group*, consists of a multiplication table of size n by n filled with numbers in the range from 1 to n. The total size is $n^2 \log n$ bits. A set of elements $S = \{g_1, \ldots, g_k\}$ of G is a *generating set* for a subgroup $H \leq G$ if every $g \in H$ can be expressed as a product of elements from S, we also write $\langle S \rangle = H$.

The *direct product* of two groups $G = (\Omega, \cdot)$ and $H = (\Omega', *)$, denoted $G \times H$, is a group with element set $\{(g, h) \mid g \in G, h \in H\}$ and an operation $\circ$ defined elementwise $(g, h) \circ (g', h') = (g \cdot g', h * h')$.

A group is *commutative* or *abelian* if for all $g, h \in G: \quad gh = hg$ holds. An abelian group is isomorphic to the direct product of cyclic groups.

A subgroup H of G is said to be *normal* if for all $g \in G$, $gH = Hg$ and we write $H \lhd G$. H is a *minimal normal subgroup* of G if there is no other normal subgroup of G contained in H. A group is *simple* if it does not have non-trivial normal subgroups. A group is *semisimple* if it is the direct product of simple groups. The *socle* of G is a subgroup generated by all minimal normal subgroups, it is denoted $Soc(G)$. The socle is semisimple and it is a normal subgroup. We shall take the cosets Hg_i as the elements of a system K. We define the product in K as $(Hg_i)(Hg_j) = Hg_k$ if $g_ig_j \in Hg_k$ in G. If H is normal, then K is a group (c.f. [Hal99], p. 27). The product depends solely on the cosets and not on the choice of the representatives. K is a group which we call the *factor group* or *quotient group* of G with respect to H and we write $K = G/H$.

A *normal series* of a group G is a finite sequence of subgroups $G_1, \ldots, G_k$ with

$$\{1\} = G_k \lhd G_{k-1} \lhd \cdots \lhd G_1 = G.$$

A normal series is a *composition series* if for each i, G_{i+1} is a proper normal subgroup of G_i and each factor group G_i/G_{i+1} is simple. The Jordan-Hölder Theorem ([Hal99], Theorem 8.4.4) states that if $\{1\} = G_k \lhd G_{k-1} \lhd \cdots \lhd G_1 = G$ and $\{1\} = H_k \lhd H_{k-1} \lhd \cdots \lhd H_1 = G$ are two composition series S_1 and S_2 for G respectively, then the factor groups G_i/G_{i+1} are isomorphic to $H_{\pi(i)}/H_{\pi(i+1)}$ in some ordering π of the indices. Note, two non-isomorphic groups could have composition series with isomorphic composition factors.

Definition 2.1 *A* complete set of coset representatives with respect to S_1 *is a sequence of tuples of group elements $\vec{s} = (s_1, \ldots, s_{k-1})$ with $s_i = (a)$ (or $s_i = (a, b)$) such that a (or a, b) generate the simple factor group G_i/G_{i+1} that is cyclic (or non-abelian, respectively) and $a(, b) \in G_i \setminus G_{i+1}$, for each $i \in \{1, \ldots, k-1\}$. All these group elements are a generating set for the group G.*

The *center* of a group G is the set of elements which commute with all elements of G, i.e. $Z(G) = \{z \in G \mid \forall\ g \in G, gz = zg\}$. This set forms a commutative subgroup of G. The *commutator* of two elements $g, h \in G$ is the element $[g, h] = g^{-1}h^{-1}gh$. A *commutator subgroup* or *derived subgroup* is the group $[G,G]$ generated by all the commutators $\{[g,h] \mid g,h \in G\}$. When iterating this, we get the *derived series* with $G^{(0)} = G, G^{(n)} = [G^{(n-1)}, G^{(n-1)}]$. It is a descending normal series $G^{(r)} \lhd \cdots \lhd G^{(1)} \lhd G^{(0)} = G$. If $G^{(i+1)} = G^{(i)}$ is non-trivial then this series terminates in a *perfect group*, i.e. it equals to its own commutator subgroup. If $G^{(i)} = \{1\}$ the trivial group, then the smallest such i is called the *derived length.* If all factor groups are commutative, then G is called *solvable.*

In contrast to the derived series, the *lower* or *descending central series* is defined $G_r \lhd \cdots \lhd G_2 \lhd G_1 = G$ where $G_2 = [G,G]$ and $G_{i+1} = [G_i, G]$. Here G_{i+1} is a normal subgroup of G_i and the factor group G_i/G_{i+1} is cyclic, for all i. If the lower central series terminates in the trivial group $G_i = \{1\}$ then G is *nilpotent.* Nilpotent groups are solvable, the converse does not hold. The smallest such i defines the *nilpotency class of* G.

The *conjugacy class* of a group element $g \in G$ is the set $cl(g) = \{h^{-1}gh \mid h \in G\}$. The *normal closure* of a group element $g \in G$ is the group $ncl_G(g) = \langle cl(g) \rangle$, i.e. a normal subgroup of G generated by all elements of the conjugacy class $cl(g)$. Note, $\mathsf{ncl}_G(g)$ is the smallest normal subgroup of G that contains g.

A *permutation* is a bijective mapping among elements in Ω. The set of all permutations together with composition forms a group, the *symmetric group* $Sym(\Omega)$. An *automorphism* ϕ is a permutation over group elements (i.e. $\phi \in Sym(\Omega)$ where $\Omega = \{g \mid g \in G\}$) such that $\phi(g)\phi(h) = \phi(gh)$. The *automorphism group* $Aut(G)$ of a group G is a group with automorphisms as elements.

Graphs. A *graph* G is a pair (V,E) with a set of *vertices* $V = V(G)$ and *edges* $E = E(G) \subseteq V \times V$. We consider *simple* graphs, i.e. with undirected edges and without loops and multiedges. The *size* of a graph is the number of its vertices. The *distance* between two vertices u, v in a graph G is the length of the shortest path between u and v.

The *degree* or *valence* of a vertex v in a graph G is the number of edges which have v as end vertex. The *valence of a graph* is the maximum valence of its vertices.

A graph is *connected* if there is a path between any two vertices. A graph is a *tree* if it is connected and does not have a simple cycle. A *root* of a tree is a designated vertex. A *rooted tree* is a tree with a root. Let $(u, v, \ldots, r)$ be a simple path from u to the root r in a rooted tree. Then v is the *parent* of u and u is a *child* of v.

An *isomorphism* between graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ is a bijective mapping $\phi : V \to V$ such that $\{u,v\} \in E_1$ if and only if $\{\phi(u), \phi(v)\} \in E_2$. Both graphs are *isomorphic* ($G_1 \cong G_2$) if such an isomorphism exists. An *automorphism* of graph G is a permutation $\phi : V(G) \to V(G)$ preserving the adjacency relation: $\{u,v\} \in E(G) \Leftrightarrow \{\phi(u), \phi(v)\} \in E(G)$. A *rigid* graph has no automorphisms except the identity.

Complexity. For the following and further complexity theoretic notions we refer to standard textbooks, for example [Pap94]. The class P (or NP) contains languages accepted by a (non-) deterministic Turing machine with polynomial time bound. The class L (or NL) contains the languages accepted by a (non-) deterministic Turing machine where the work-tape is restricted to $O(\log n)$ bits. Non-determinism means, that the machines are allowed to *guess* bits while computing the solution and verrify it within the restricted resource bounds. The class AC^i contains the languages accepted by a DLOGTIME uniform family of Boolean circuits of depth $O(\log^i n)$ and size polynomial in n, with unbounded fanin *and*-gates and *or*-gates. The class NC^i is AC^i where *and*-gates and *or*-gates have bounded fanin (i.e. fanin two). $\mathsf{NC} = \bigcup_i \mathsf{NC}^i$. The class SAC^i is AC^i but with bounded fanin *and*-gates. SAC^1 is also known as LogCFL. The following containments are known:

$$\mathsf{AC}^0 \subset \mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{SAC}^1 \subseteq \mathsf{AC}^1 \subseteq \mathsf{NC}^2 \subseteq \mathsf{SAC}^2 \subseteq \mathsf{AC}^2 \subseteq \cdots \subseteq \mathsf{NC} \subseteq \mathsf{P} \subseteq \mathsf{NP}$$

A language L is AC^0 *many-one reducible* (in short $\leq_m^{\mathsf{AC}^0}$) to a language L' if there is a total function f computable in AC^0 so that for all words, $w \in L_1$ if and only if $w \in L_2$. We consider reducibility with respect to $\mathsf{DLOGTIME}$ uniformity. The notions L, NC or P many-one reducibility are defined accordingly.

3 Reduction to Graph Isomorphism

In this section we show how group isomorphism can be reduced to graph isomorphism. Following the reduction in [Mil79], the valence of the graphs is not bounded. We introduce a different reduction to graph isomorphism. If we break up the group into smaller pieces then in our construction the valence of the constructed graphs can be reduced.

The construction. Let G be a group with n elements in table representation. The reduction goes in two steps.

First, we define a graph $T(G)$ as follows: For every group element $g \in G$ there is an *element vertex* g in $T(G)$. There is a *root vertex* eG, it is connected to all element vertices in $T(G)$.

Second, from this tree we construct a graph $X(G)$. $X(G)$ contains a main copy of $T(G)$. For every element vertex g in $T(G)$, we have a further copy T_g of $T(G)$. The root of T_g is identified with g. The leafs of T_g are connected to graph gadgets which encode the multiplication in G. Figure 1 shows the construction for a multiplication graph gadget $M_{gh=k}$. Hence, each node $g^{(h)}$ is connected to three multiplication graph gadgets, namely when g is multiplied with h to the left, i.e. to $M_{gh=k}$, and if to the right, i.e. to $M_{hg=i}$, and when g is the result of a multiplication with h from right $M_{jh=g}$, for corresponding group elements $i, j \in G$.

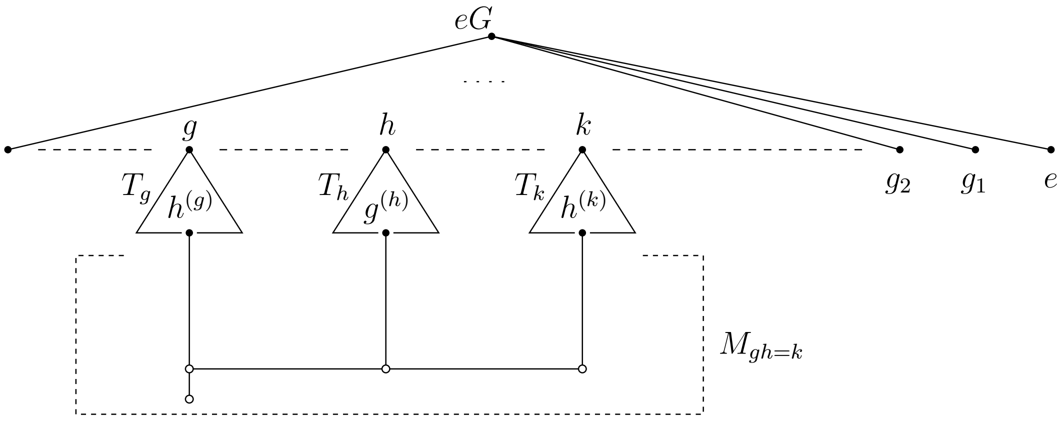

Figure 1: The graph $X(G)$ with element vertices e, g_1, g_2, g, h, k is shown. A multiplication graph gadget $M_{gh=k}$ is indicated by white nodes.

Claim 3.1 *Every automorphism in G can be directly transformed into an automorphism in $X(G)$.*

Proof. To see this, the root node eG is the only node with valence n and so it is fixed. Hence, the vertices at each level in the construction are mapped onto each other. When the element vertices of the main copy $T(G)$ in $X(G)$ are fixed, then the whole graph is fixed. That is, because for every triple $g, h, k \in G$ with $gh = k$ there is a unique multiplication graph gadget and hence, there are unique paths from g to h and to k. Hence, $g^{(h)}, h^{(g)}, k^{(h)}$ are fixed. Since every vertex at distance two to the root eG is connected to at least one multiplication graph gadget, all nodes of this level are fixed. The multiplication graph gadgets are rigid. $\square$

Refinement of groups. The graph $X(G)$ has vertices with valence at least n. We construct a new tree $T'(G)$. Let $X'(G)$ be the graph $X(G)$ where we replace each copy of $T(G)$ by $T'(G)$. The goal is that in $X'(G)$ the nodes have smaller valence but the automorphism properties of $X'(G)$ are the same as in $X(G)$.

Let N be a normal characteristic subgroup in G. We get a normal series, namely $\{1\} \lhd N \lhd G$. Every group element $g \in G$ can be written as a product $h_1 n$ where $n \in N$ and $h_1 \in h_1 N$ is a coset representative. Let $h_1, \ldots, h_k$ be a complete set of coset representatives.

We define a new tree $T'(G)$ as follows.

- For every group element $g \in G$ with $g = h_i n$ ($i \in \{1, \ldots, k\}$) there is an *element vertex* $h_i n$ in $T'(G)$.
- For every coset representative $h_1, \ldots, h_k$ there is a vertex $h_i N$ in $T'(G)$.
- There is a *root vertex* eG connected to the vertices $h_1 N, \ldots, h_k N$ in $T'(G)$.

The rest of the construction of $X'(G)$ is the same as for $X(G)$. In particular, the tree copies connected to the leafs of $T'(G)$ are copies of $T'(G)$.

Remarks. We use the property that N is characteristic and that the cosets $N, h_1 N, \ldots, h_k N$ are mapped blockwise onto each other. The tree $T'(G)$ exactly has these automorphisms. The valence of the graph is now the maximum of $1 + k$ and $1 + n/k$ where k is the number of cosets and $|N| = n/k$. Note, here we neglect that e.g. $g^{(h)}$ has at least 4 neighbours. We remedy this later with minor changes to the construction. Figure 2 shows an example.

Another important point is, that this construction does not depend on the selection of the coset representatives. For example, take $h_1' \in h_1 N$ instead of h_1 as coset representative. Since $h_1' \in h_1 N$ there exists $n \in N$ such that $h_1' = h_1 n$ or equivalently $h_1 = h_1' n^{-1}$. Every element $h_1 n'$ can be written as $(h_1' n^{-1}) n' = h_1'(n^{-1} n')$. Hence, taking a different element as coset representative implies a rearrangement among the children of the node $h_1 N$ in $T'(G)$. Also if $h_1' \in h_i N$ instead of $h_1 N$ then this means that we also permute the cosets. This implies a rearrangement among the nodes $h_1 N, \ldots, h_k N$ in $T'(G)$.

We repreat the decomposition of N and the factor group G/N to further reduce the valence. That is, in the tree $T'(G)$ we add new intermediate layers. Hence, we end up in a composition series. The valence depends on the size of the largest factor group in the decomposition. In a composition series the factor groups are simple, then it suffices to take at most two elements to define coset representatives for all cosets. For example, for a cyclic factor group of size l, we have $eN, gN, g^2 N, \ldots, g^{l-1} N$. This is shown for p-groups in Section 5 and generalized to groups in Section 6. The theory we need for the decomposition of groups is provided in Section 4.

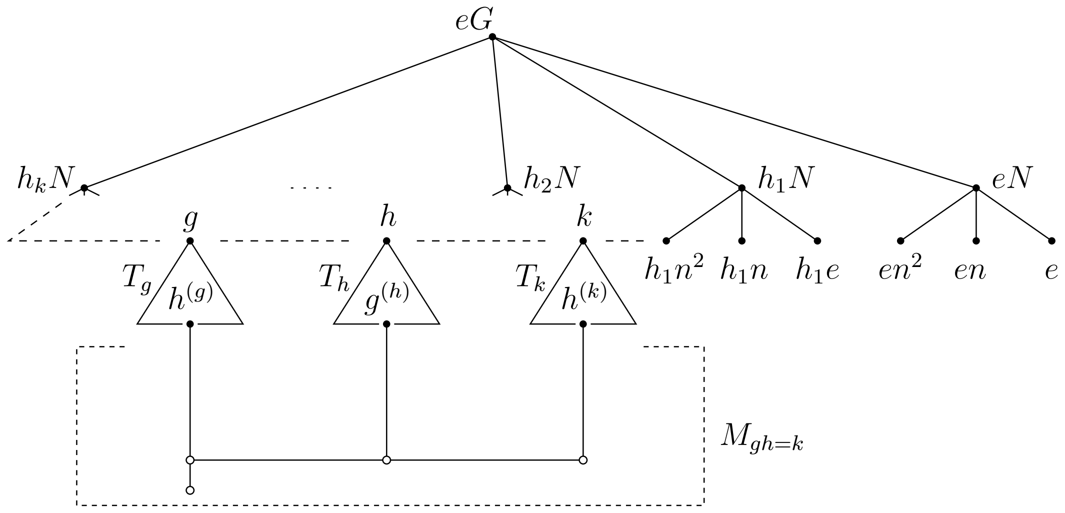

Figure 2: The graph $X'(G)$ is shown with coset representatives $e, h_1, \ldots, h_k$ and with normal subgroup N, a cyclic group of three elements. The labels of the element vertices are written as products with an element in N and a coset representative.

4 Decomposition of Groups

In this section we describe how groups can be decomposed. The recursive decomposition of groups into normal subgroups and factor groups ends up in simple groups. Here, we just distinguish between two types of simple groups, namely the cyclic groups and non-abelian simple groups. In our algorithms we use the following fact.

Theorem 4.1 *Finite simple non-abelian groups can be generated by two group elements.*

The proof of this theorem depends on the classification of the finite simple groups. Clearly, a cyclic group of prime order can be generated by one element. The non-abelian simple groups can be generated by two elements where one is an involution (c.f. [MSW94]).

In the following we discuss two points. First, when decomposing groups it would be useful to reduce the number group elements we need to define a complete set of coset representatives. For example if we have h_1 as a representative for h_1N, then we can take h_1^2 as a representative for the coset h_1^2N. The goal is to describe all coset representatives by a small set of generators, namely those elements that generate the factor group G/N. If the factor groups are simple, then we have at most two elements for this. We show now, that all group elements in a simple group can be arranged uniquely when one or two generators are fixed.

Second, we show how the groups can be decomposed.

4.1 Arrange group elements by generators

We show how group elements can be arranged uniquely with respect to a set of generators.

Generator-representations for groups. For a group G there is a composition series S where the factor groups G_i/G_{i+1} for $i \in \{1, \ldots, k-1\}$ are simple groups, i.e. cyclic of prime order for a prime p_i, or simple and non-abelian and generated by two elements:

$$\{1\} = G_k \lhd G_{k-1} \lhd \cdots \lhd G_1 = G$$

where $\prod_{i=1}^{k-1} p_i = n$.

- If G_i/G_{i+1} for $i \in \{1, \ldots, k-1\}$ is cyclic, then we need one generator $a_i G_{i+1}$ for G_i/G_{i+1}.
- If G_i/G_{i+1} is non-abelian and simple, then we need two generators $a_i G_{i+1}, b_i G_{i+1}$ for G_i/G_{i+1}.

Let S be given by $\vec{s} = (s_1, \ldots, s_{k-1})$, we say that an element g is given in *generator-representation*, if it can be expressed as a product $w_1 \cdots w_{k-1}$ where

- $w_i = a_i^{l_i}$ with $s_i = (a_i)$ and $l_i \in \{0, \ldots, p_i - 1\}$ if G_i/G_{i+1} is cyclic,
- w_i is a uniquely determined word with generators a_i, b_i with $s_i = (a_i, b_i)$ if G_i/G_{i+1} is non-abelian and simple.

Unique order for group elements. The following lemma sais how group elements in simple groups can be arranged uniquely according to their generator-representation.

Lemma 4.2 *Let G be a group and S a composition series given by $\vec{s}$ as in Definition 2.1. There is a logspace computable function that brings every group element in G into a new order that is uniquely determined according to their generator-representation.*

Proof. Let $\vec{s} = (s_1, \ldots, s_{k-1})$. The group elements are arranged in lexicographical order according to their generator representation, i.e. with highest priority sort group elements according to a word w_1 of group elements in s_1, then those which are equal are sorted according to a word w_2 of group elements in s_2 and so on, until we sort according to a word w_{k-1} of group elements in s_{k-1}.

Clearly, if $s_i = (a)$ (i.e. G_i/G_{i+1} is cyclic) then a word w_i is a power of a, these can be distinguished by their exponent $a^0 < a^1 < \cdots < a^{\mathsf{ord}(a)-1}$. If $s_i = (a, b)$ (i.e. G_i/G_{i+1} is non-abelian and simple) and we have an order $a < b$, then we compare products of these generators as words w_i lexicographically. Hence, it remains to say how to get unique words for the elements in this factor groups.

Consider the Cayley-graph of G_i/G_{i+1} where we have directed edges labeled with a_i, a_i^{-1} and b_i, b_i^{-1}. Since a_i, b_i are generators, this graph is connected. We define as order $a_i < a_i^{-1} < b_i < b_i^{-1}$.

Claim 4.3 *In a Cayley-graph, if all generators are arranged in a unique order then there is a logspace computable function which arranges all group elements in a unique order.*

Proof. The proof is adapted from [DLN08], where planar 3-connected graphs embedded in a plane are canonized. This procedure also works in logspace for general graphs when given a cyclic arrangement for the edges going around each vertex, c.f. *oriented graphs* in [Wag10].

Let $a_1 < a_1^{-1} < \cdots < a_{k-1} < a_{k-1}^{-1}$ be a unique order for all generators. We complete this to a cyclic arrangement ρ with $a_{k-1}^{-1} < a_1$. Hence, the cayley graph is an oriented graph now. We follow a path starting at node e in direction of the edge labeled with a_1 with a *universial exploration sequence* [Rei08]. That is, a logspace machine traverses the whole graph and reaches in polynomial number of steps every vertex at least once. Let p be this path. A second logspace machine goes through this path p and computes all the positions when each vertex is reached for the first time. These paths up to a certain position can be seen as a product with generators that evaluates to a group element. We can sort all group elements according to the position of their first occurence in this path. □

This completes the proof of Lemma 4.2. □

Since the computations are deterministic and independent from the table representation of the group elements, we immediately get the following corollary.

Corollary 4.4 *Let G, H be two groups and S, S' be composition series given by complete sets of coset representatives. If these induce an isomorphism from G onto H then Lemma 4.2 gives that isomorphism from G onto H.*

4.2 Computation of a composition series.

Let G be a group with n elements. We decompose G into a normal subgroup N (the socle of G) and a factor group G/N. We do this recursively until we end up in a *composition series* where each factor group is semisimple (i.e. a direct product of simple groups).

First, we compute the normal closure $\mathsf{ncl}(x)$ for all elements $x \in G$. Note, $\mathsf{ncl}(x) = G$ for all $x \in G$ if and only if G is already simple (i.e. cyclic if dealing with solvable groups). The algorithm will give us a generator, namely for $i \in \{1, \ldots, k-1\}$ and $a_i = x$ we get $a_i G_{i+1}$ as generator for G_i/G_{i+1}.

Note, $ncl(x)$ is the smallest normal subgroup which contains x. These are minimal normal subgroups, i.e. a direct product of isomorphic simple groups. We run through all elements $x \in G$ and select those, where $ncl(x)$ is not contained in another subgroup $ncl(y)$. It is also useful to have the following (c.f. Proposition 1.5.1 in [Faw09]).

Fact 4.5 *Any two distinct minimal normal subgroups of a group G must intersect trivially.*

The socle $Soc(G)$ is a subgroup generated by all minimal normal subgroups. The socle is a normal subgroup of G, it is a direct product of simple groups. For these and more facts, see [Faw09].

There is one task, namely that we have to break up the socle into its simple groups. For this we use Proposition 4.5, namely that the simple groups from the socle come from all the minimal normal subgroups. Hence, it suffices to break up each minimal normal group N into its simple groups:

- If N is abelian, then consider all elements of prime order. Go through them from left to right and select an element as generator if it is not generated by the group elements of prime order to the left.

  To see this, we refer to Lemma 3.2.1 and 3.2.2 in [Hal99], namely that every group element can be written uniquely as a product of prime power elements, and prime power elements can be generated by prime elements.

- If N is not abelian, then the simple groups in N are not abelian. We need two generators. We search a pair of elements in N which generates a simple group K_1 which is normal in N. For this, we compute $\mathsf{ncl}_N(g)$ for $g \in G$ until we find a non-trivial normal subgroup K_1 of N. We do this recursively for N/K_1. Hence, we get generators for all simple groups $K_1, \ldots, K_i$ in $N = K_1 \times \cdots \times K_i$.

Arrange factor groups in composition series. The socle of a group is semisimple, it is generated by the minimal normal subgroups. These are pairwise disjoint, i.e. their intersection contains $\{e\}$ only. Minimal normal subgroups are a direct product of isomorphic simple groups. Hence, the recursive decomposition of groups into normal subgroups and factor groups is not unique. For example the cyclic group C_6 has two different composition series:

$$\{1\} \lhd C_2 \lhd C_6 \text{ and } \{1\} \lhd C_3 \lhd C_6.$$

Here, the factor groups are C_2 and C_3 which appear in different order. The following Theorem sais, that the factor groups are isomorphic. But this does not mean that groups are isomorphic if they have isomorphic factor groups.

Theorem 4.6 *(Jordan-Hölder) Two composition series of a group have isomorphic composition factors.*

For an isomorphism test, we arrange the simple groups of the socle. This can be done as follows.

In the case of abelian groups isomorphism testing can be done in linear time [Kav03]. In the algorithm, the orders of all group elements are simply compared. Hence, when sorting these sequences they can be compared lexicographically to define an order $\prec$ on them.

We define an order $G \prec H$ on two simple non-abelian groups if one of the following holds:

- $|G| < |H|$ or
- $|G| = |H|$ but $T_G < T_H$ which is computed as follows. Let (a', b') be a pair of elements which generate H. Since we have two generators only for simple groups, we run through all pairs (a, b) of group elements in G. As in Claim 4.3 we compute Cayley graphs with $a < b$ in G and $a' < b'$ in H. The claim sais, that the group elements can be arranged in a unique order in logspace. Now, compare the multiplication tables T_G and T_H with group elements arranged in this order line by line and bit by bit.

We define $G = H$ if the groups are isomorphic, i.e. if neither $G \prec H$ nor $H \prec G$ holds. We define $G \preceq H$ if $G \prec H$ or $G = H$ holds.

With this order we define an order on the composition factors in a composition series of a group.

Definition 4.7 *A* sorted sequence of composition factors *is a sequence of factor groups in a composition series where factor groups are sorted as follows.*

1. *According to the recursive decomposition into normal subgroups and factor groups, i.e.* $\{1\} \lhd Soc(G) \lhd G$*, where we refine* $Soc(G)$ *and* $G/Soc(G)$ *recursively.*
2. *The socles in the decomposition process are direct products of simple groups, i.e.* $Soc(G) = K_1 \times \cdots \times K_i$*. Arrange* $K_1, \ldots, K_i$ *according to* $\prec$*, for example if* $K_1 \preceq \cdots \preceq K_i$ *then* $\{1\} \lhd K_1 \lhd \cdots \lhd K_{i-1} \lhd Soc(G)$*. Among them, arrange isomorphic simple factor groups arbitrarily.*

For a composition series S, let $\mathsf{seq}(S)$ be the sequence of composition factors in S. We assume that the coset representatives in $\vec{s}$ are arranged as in $\mathsf{seq}(S)$.

We prove that for an isomorphism test, it suffices to arrange isomorphic simple factor groups in an arbitrary order. In the isomorphism algorithm we run through all possibilities to select coset representatives. Hence, this also includes the case of interchanging the here selected coset representatives of isomorphic factor groups.

In arbitrary two composition series the composition factors are ordered with respect to Step 1 in Definition 4.7. That is because the socle of a group is a *characteristic* subgroup. According to Step 2, we arrange the factor groups with respect to the order $\prec$. This can be done, because the groups $K_1, \ldots, K_i$ are all normal subgroups in G.

Lemma 4.8 *For two isomorphic groups G and H, there are composition series S for G and S' for H as in Definition 4.7 with $(G, \mathsf{seq}(S))$ is isomorphic to $(H, \mathsf{seq}(S'))$ such that this isomorphism mapps the factor groups in S blockwise onto the factor groups in S'.*

The algorithm. In Algorithm 1 we show how a composition series is computed according to Lemma 4.8. We give then some comments to the three main parts of the algorithm.

Algorithm 1 CompSeries: Compute composition series for groups.

```
input: group $G$ with $n$ elements
output: composition series for $G$ given by a complete set of coset representatives
initialize: set of groups $NCL = \{\}$
              { 1. compute minimal normal subgroups }
 1: for each $g \in G$ do
 2:    $\mathsf{ncl}_G(g) = \langle g_1, \ldots, g_k \rangle$ with $\{g_1, \ldots, g_k\} = \{h^{-1}gh \mid h \in G\}$
 3:    if $\exists N \in NCL :\ \mathsf{ncl}_G(g) < N$ then $NCL = \{\mathsf{ncl}_G(x)\}$ else $NCL \leftarrow ncl_G(x)$
 4: end for
 5: $Soc(G) = \langle N_1, \ldots, N_k \rangle$ with $\{N_1, \ldots, N_k\} = NCL$
 6: if $\forall g \in G :\ \mathsf{ncl}_G(g) = G$ then $G$ is simple, return $\{G\}$
              { 2. compute simple groups }
 7: for each $N \in NCL$ do
 8:    if $N$ is abelian then
 9:       for each $\{g \in N \mid \mathsf{ord}(g)$ prime, $g \notin \langle S(N) \rangle\}$ do $S(N) \leftarrow (g)$
10:    if $N$ is not abelian then
11:       for each $\{(g,h) \in N \times N \mid g \neq h,\ \exists n \in N :\ \langle g,h \rangle = \mathsf{ncl}_N(n)\}$ do
12:          if $\langle g,h \rangle \cap \langle S(N) \rangle = \{e\}$ then $S(N) \leftarrow (g,h)$
13: end for
              { 3. compute all composition series }
14: $\vec{s}_1 = \mathsf{CompSeries}(G/Soc(G))$ where $G/Soc(G) = \{gSoc(G) \mid g \in G \setminus (Soc(G) \setminus \{e\})\}$
15: choose arbitrarily $(N_1, \ldots, N_k) \in Sym(NCL)$ with $s_i \preceq s_j$ for all $i < j, s_i \in N_i, s_j \in N_j$
16: choose arbitrarily $(s_{1,1}, \ldots, s_{1,l_1}, \ldots, s_{k,1}, \ldots, s_{k,l_k}) \in Sym(S(N_1)) \times \cdots \times Sym(S(N_k))$
17: $\vec{s}_2 = (s_{1,1}, \ldots, s_{1,l_1}, \ldots, s_{k,1}, \ldots, s_{k,l_k})$ represents the socle, i.e.
          $Soc(G) = G_{1,1} \lhd G_{1,2} \lhd \cdots \lhd G_{k,l_k} \lhd G_{k,l_k+1} = \{1\}$
          with $G_{i,j}/G_{i,j+1} = \langle aG_{i,j+1} \rangle$ if $s_{i,j} = (a)$ and
          with $G_{i,j}/G_{i,j+1} = \langle aG_{i,j+1}, bG_{i,j+1} \rangle$ if $s_{i,j} = (a,b)$
18: $\vec{s} = (s_1, \ldots, s_i, s_{i+1}, \ldots, s_j)$ with $(s_1, \ldots, s_i) = \vec{s}_1, (s_{i+1}, \ldots, s_j) = \vec{s}_2$
19: return $\vec{s}$
```

In the first part, we compute the minimal normal subgroups and put them into the set NCL. The socle $Soc(G)$ is then generated by all members in NCL. If $ncl(g) = G$ for all $g \in G$ then G is simple, the composition series is trivially $\{1\} \lhd G$.

In the second part, we compute for each minimal normal subgroup N its simple groups. We distinguish the situation whether N is abelian or not. Then we need one or two generators for a simple group, respectively.
In Line 9 and 12, we write in short $\langle S(N) \rangle$ and mean the group generated by all group elements a and b with $(a) \in S(N)$ or $(a, b) \in S(N)$.
In Lines 10 to 12, we run through all pairs $g, h \in N$ until we find one which generates N. Recall, that a direct product GH of simple groups G and H has normal subgroups, namely G and H, and $G \cap H = \{e\}$.

In the third part, we compute $\vec{s}_1$ we encode a composition series for the factor group $G/Soc(G)$ recursively. In $\vec{s}_2$ we encode a composition series for $Soc(G)$. We take the generators (i.e. in the tuples $s_{i,j}$ for $i \in \{1, \ldots, k\}, j \in \{1, \ldots, l_i\}$) of all the composition factors as coset representatives and get a complete set of coset representatives for G. The factor groups are arranged according to the recursive decomposition and then according to $\prec$ as in Definition 4.7. In Line 18, $\vec{s}_2$ is the same as in Line 17, the elements are just relabeled.

Each step can be computed in polynomial time. We get the following lemma.

Lemma 4.9 *Let G, H be two isomorphic groups. A composition series S for G and S' for H such that $(G, \mathsf{seq}(S))$ is isomorphic to $(H, \mathsf{seq}(S'))$ can be computed in polynomial time.*

Remarks. When analyzing the complexity of each line in the algorithm. It can be verrified that each step can be done in logspace if the computations of the previous steps are given in the input: The main tasks include for example set operations, computing minimum, evaluating products of group elements. Note, that computing the group generated by a given set of elements can be reduced to reachability testing queries in the corresponding Cayley graph when considering edges to be undirected.
In Line 14, the recursion depth is at most $O(\log n)$. The computation for one step in the recursion can be done by an SAC^1 circuit.

Corollary 4.10 *Let G, H be two isomorphic groups. A composition series S for G and S' for H such that $(G, \mathsf{seq}(S))$ is isomorphic to $(H, \mathsf{seq}(S'))$ can be computed in* SAC^2.

5 Isomorphism for p-groups

In a composition series for a p-group every composition factor is cyclic of prime order p. We use this fact and reduce group isomorphism where groups are given as in Definition 4.7 to isomorphism testing on graphs of valence at most $p + 1$.

In this section we assume, that a sequence of composition factors is given with respect to a composition series S for a group G and that for S we have a complete set of coset representatives $\vec{s}$. We denote the sequence of composition factors of S by $\mathsf{seq}(S)$. Recall, any element in a coset could be taken as a representative. So an isomorphism from $(G, \mathsf{seq}(S))$ onto $(H, \mathsf{seq}(S'))$ just mapps cosets blockwise onto each other.

5.1 p-Groups and Graphs of Bounded Valence

In this section we give a reduction from p-group isomorphism onto graph isomorphism with graphs of valence $p+1$. First, we repeat the notation of the given groups. For a prime number p, let (G,S) be a p-group G over n elements with a composition series S where each factor group G_i/G_{i+1} is cyclic of order p as in Definition 4.7:

$$\{1\} = G_k \lhd G_{k-1} \lhd \cdots \lhd G_1 = G$$

and let $\vec{g} = ((g_1), \ldots, (g_{k-1}))$ be a complete set of coset representatives with respect to S with $g_i G_{i+1} \in G_i/G_{i+1}$. Let (H,S') be a p-group H with a composition series S' where each factor group H_i/H_{i+1} is cyclic of order p as in Definition 4.7:

$$\{1\} = H_k \lhd H_{k-1} \lhd \cdots \lhd H_1 = H$$

and let $\vec{h} = ((h_1), \ldots, (h_{k-1}))$ be a complete set of coset representatives with respect to S' with $h_i H_{i+1} \in H_i/H_{i+1}$.

Theorem 5.1 *There is an AC^0-computable function that on input of groups G, H and S, S' computes a graph $X_p(G,S)$ and $X_p(H,S')$ with at most $11n^2+1$ vertices which have valence at most $p+1$ such that $X_p(G,S)$ is isomorphic to $X_p(H,S')$ if and only if $(G, \mathsf{seq}(S))$ is isomorphic to $(H, \mathsf{seq}(S'))$, i.e. there exist composition series such that their subgroups are isomorphic to G_i and H_i and these are mapped onto each other blockwise, for all $i \in \{1, \ldots, k-1\}$.*

Proof. We construct a graph $X_p(G,S)$ for G and S in two steps. The construction for $X_p(H,S')$ is done accordingly.

In the first step we define a tree $T_p(G,S)$. Intuitively speaking, this tree is based on the structure of the composition series S.

1. For every group element $g \in G$ we have an *element vertex* in $T_p(G,S)$.
2. For all $i \in \{1, \ldots, k-1\}$ and each coset gG_i and subgroup eG_i of the factor group G/G_i we have a *coset vertex* gG_i and eG_i in $T_p(G,S)$. Note, for $i=1$ we call $eG_1 = eG$ the *root* of $T_p(G,S)$.
3. For each element vertex g and coset vertex gG_{k-1} we have an edge $\{g, gG_{k-1}\}$ in $T_p(G,S)$.
4. For each pair of coset vertices $gG_i, g'G_{i+1}$ with $g, g' \in gG_i$ we have an edge $\{gG_i, g'G_{i+1}\}$ in $T_p(G,S)$.

Now we construct $X_p(G,S)$. We use $T_p(G,S)$ and define a further graph gadget to simulate the multiplication rule.

1. We have a main copy of $T_p(G,S)$ in $X_p(G,S)$. The root node of the main copy is connected to a *color graph gadget*, namely a path of length two to distinguish this vertex from the others. For each leaf node v in $T_p(G,S)$ we have a copy T_v of $T_p(G,S)$. We identify the root node of T_v with v.
2. For each node v in $T_p(G,S)$, we have for each leaf node w of T_v five nodes $w_\leftarrow, w_\rightarrow, w_l, w_r, w_=$ in $X_p(G,S)$. We have edges $(w, w_l), (w_l, w_\leftarrow), (w, w_r), (w_r, w_\rightarrow), (w_r, w_=)$ in $X_p(G,S)$.

3. For each pair of group elements $g, h \in G$ we simulate the multiplication $gh = k$ as follows. We define a *multiplication graph gadget* that is connected to the vertices $h^{(g)}_{\leftarrow}$ in T_g, $g^{(h)}_{\rightarrow}$ in T_h, and $h^{(k)}_{=}$ in T_k. One gadget $M_{gh=k}$ is shown in Figure 3.

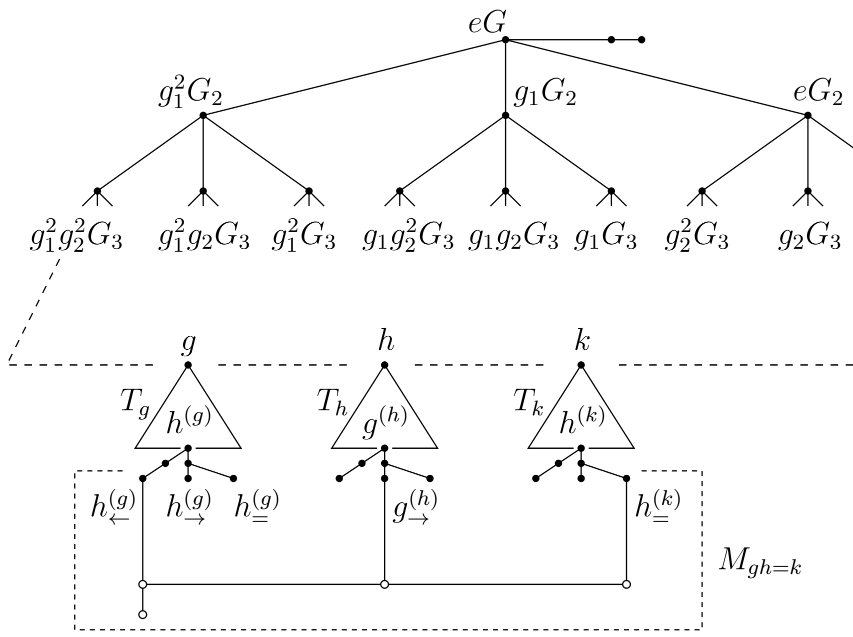

Figure 3: The graph $X_p(G, S)$ and a multiplication graph gadget $M_{gh=k}$ indicated by white nodes.

We prove now that $X_p(G, S)$ has all the properties stated in Theorem 5.1.

Claim 5.2 *The graph $X_p(G, S)$ has at most $11n^2 + 1$ vertices.*

Proof.

- $T_p(G, S)$ is a complete p-ary tree with n leaves, it has $\sum_{i=1}^{\log_p(n)-1} p^i \leq 2n - 1$ vertices.
- The color graph gadget is a path of length two connected to the root node of $T_p(G, S)$.
- For each of the n leaf nodes of $T_p(G, S)$ we have a copy of $T_p(G, S)$. Since the root nodes of these copies are identified with the leaf nodes of $T_p(G, S)$ we do not count these nodes twice. We get at most $n((2n - 1) - 1)$ vertices.
- Every leaf node v of these copies is connected to a subtree with five nodes $v_l, v_r, v_{\leftarrow}, v_{\rightarrow}$ and $v_{=}$. There are n^2 such leaf nodes. Hence, we get further $5n^2$ vertices.
- Every multiplication gate has 4 vertices, when not counting the vertices $v_{\leftarrow}, v_{\rightarrow}$ and $v_{=}$ twice. We have n^2 multiplication gates. We get $4n^2$ vertices.

We have in total $2n - 1 \quad + \quad 2 \quad + \quad n((2n - 1) - 1) \quad + \quad 5n^2 \quad + \quad 4n^2 = 11n^2 + 1$ vertices. □

Claim 5.3 *There is a logspace-computable function that computes the graph $X_p(G, S)$.*

Proof. Since we have coset representatives $(g_1, \ldots, g_{k-1})$, every element $gG_i \in G_1/G_i$ can be obtained uniquely by following a path

$$(eG, g_1^{l_1}G_2, g_1^{l_1}g_2^{l_2}G_3, \ldots, \quad g_1^{l_1} \cdot \cdots \cdot g_{i-1}^{l_{i-1}}G_i)$$

this is related to the following product:

$$gG_i = g_1^{l_1}g_2^{l_2} \cdot \cdots \cdot g_{i-1}^{l_{i-1}}G_i$$

The logspace machine goes through all products in lexicographically increasing order to the sequence of exponents, i.e. in the above example: $(l_1, \ldots, l_{i-1})$. The construction of the tree is done as in a depth first traversal through the resulting tree $T_p(G, S)$. We evaluate the products for all group elements $g = g_1^{l_1} \cdots g_{k-1}^{l_{k-1}}$ for $0 \leq l_i \leq p - 1$ (for all i) in a preprocessing step. This can be done in logspace. A further logspace machine relabels the group elements by such products, i.e. we rewrite the whole group table.

It is easy to see, that with access to the new group table each step of the graph construction can be done in AC^0. $\square$

Claim 5.4 *The graph $X_p(G, S)$ has valence at most $p + 1$ for all $p \geq 2$.*

Proof. The nodes in the tree $T_p(G, S)$ have one parent and p children, because in Step 4 of the construction, there are at most p cosets $g'G_{i+1}$ in gG_i with $g, g' \in gG_i$. The root node is connected to a color graph gadget and has also valence $p + 1$. The leafs are identified with the root node of a copy of $T_p(G, S)$ and have also valence $p + 1$. Each leaf node of the copies of $T_p(G, S)$ is connected to at most one multiplication graph gadget and has valence two. To see this we argue, that for every pair of variables in $gh = k$ the third variable is uniquely determined in a group. The vertices of the multiplication graph gadgets and the color graph gadget have valence at most 3, this is not greater than $p + 1$ for all $p \geq 2$. $\square$

Claim 5.5 *The graph $X_p(G, S)$ is isomorphic to $X_p(H, S')$ if and only if $(G, \mathsf{seq}(S))$ is isomorphic to $(H, \mathsf{seq}(S'))$.*

Proof. Consider an isomorphism ϕ between $(G, \mathsf{seq}(S))$ and $(H, \mathsf{seq}(S'))$, we argue now that we get an isomorphism between $X_p(G, S)$ and $X_p(H, S')$. That is, for every pair of elements g, h in G we show that the property $\phi(g)\phi(h) = \phi(gh)$ can be transformed into an isomorphism between the graphs.

Clearly, an isomorphism from $(G, \mathsf{seq}(S))$ onto $(H, \mathsf{seq}(S'))$ is also an isomorphism when just considering $T_p(G, S)$ and $T_p(H, S')$. Now, consider $X_p(G, S)$ and $X_p(H, S')$. Let $gh = k$, between three leaf nodes g, h, k of $T_p(G, S)$ in $X_p(G, S)$ there are unique simple paths going through a single multiplication graph gadget $M_{gh=k}$ such that:

- except g, h, k there is no other vertex visited in $T_p(G, S)$,
- there is a unique simple path from g of $T_p(G, S)$ to $h_{\leftarrow}^{(g)}$ in T_g,
- there is a unique simple path from $h_{\leftarrow}^{(g)}$ of T_g to $g_{\rightarrow}^{(h)}$ of T_h in M,
- there is a unique simple path from $g_{\rightarrow}^{(h)}$ of T_h to h of $T_p(G, S)$ in T_h,

- there are unique simple paths from $h_{\leftarrow}^{(g)}$ of T_g and $g_{\rightarrow}^{(h)}$ of T_h to $h_{=}^{(k)}$ in T_k in M,
- there is a unique simple path from $h_{=}^{(k)}$ of T_k to k of $T_p(G,S)$ in T_k.

Hence, if ϕ is an isomorphism of $(G, \mathsf{seq}(S))$ onto $(H, \mathsf{seq}(S'))$ then in $X_p(H,S')$ there is also a multiplication graph gadget $M_{\phi(g)\phi(h)=\phi(k)}$ such that these unique simple paths exist. This isomorphism mimics the permutation from $T_p(G,S)$ onto $T_p(H,S')$ also at each copy of the tree, e.g. T_g in $X_p(G,S)$ is mapped onto $T_{\phi(g)}$ in $X_p(H,S')$, and for every leaf vertex v with $\phi(v) = w$, the vertices $v_{\leftarrow}, v_{\rightarrow}, v_{=}$ in T_g are mapped via ϕ onto $w_{\leftarrow}, w_{\rightarrow}, w_{=}$ in $T_{\phi(g)}$.

Now to the other direction. Since the root node of $T_p(G,S)$ is distinguished from the others, any isomorphism mapps this root node onto the root node of $T_p(H,S')$. Vertices at the same distance are mapped onto each other, hence $T_p(G,S)$ is mapped onto $T_p(H,S')$. This also holds for the copies of the tree rooted at the childen and the multiplication graph gadgets.

Any isomorphism respects the multiplication rules of the groups: There are multiplication graph gadgets just for the multiplication rules, i.e. if $gh = k$ in G then there is no gadget $M_{gh=k'}$ for any $k' \neq k$. Since the multiplication graph gadget is rigid, there is no isomorphism that mapps a vertex $v_{\leftarrow}$ in $X_p(G,S)$ onto any vertex $w_{\rightarrow}$ or $w_{=}$ in $X_p(H,S')$ and vice versa. We conclude, every isomorphism ϕ from $(G, \mathsf{seq}(S))$ onto $(H, \mathsf{seq}(S'))$ mapps $M_{gh=k}$ in $X_p(G,S)$ onto $M_{\phi(g)\phi(h)=\phi(k)}$ in $X_p(H,S')$. Hence, if $(G, \mathsf{seq}(S))$ is not isomorphic to $(H, \mathsf{seq}(S'))$ then we cannot get an isomorphism from $X_p(G,S)$ onto $X_p(H,S')$.

There is a one-to-one correspondence between automorphisms of $X_p(G,S)$ and automorphisms of $(G, \mathsf{seq}(S))$. Assume, the leaf nodes of $T_p(G,S)$ are fixed. Since any three leaf vertices of $T_p(G,S)$ have at most one rigid multiplication graph gadget in common, all of them are fixed. Since every vertex $w_{\leftarrow}^{(v)}, w_{\rightarrow}^{(v)}, w_{=}^{(v)}$ of every tree T_v in $X_p(G,S)$ is connected to a multiplication graph gadget, all these vertices are fixed. We conclude, that every automorphism of $(G, \mathsf{seq}(S))$ induces a unique automorphism of $X_p(G,S)$. Hence, this also holds for isomorphisms from $X_p(G,S)$ onto $X_p(H,S')$. $\square$

This completes the proof of Theorem 5.1. $\square$

Note, the graph $X_p(G,S)$ has the property that it is a *cone-graph* with logarithmic depth bound. That is, from every vertex, there is a unique path to the root node of $T_p(G,S)$.

Complexity. For an isomorphism test, we compute first a composition series S and S' and then the sequences of composition factors $\mathsf{seq}(S)$ and $\mathsf{seq}(S')$, we sort them as in Definition 4.7. This can be done in polynomial time by Lemma 4.9 Also the complexity of bounded valence GI is in polynomial time:

Theorem 5.6 *([BL83]) Isomorphism on graphs of valence at most d can be tested in deterministic time $n^{O(d)}$.*

We put this together and get for a constant c the running time $n^c + n^{c(p+1)}$. This completes the proof of Theorem 1.2 and Theorem 1.3.

Theorem 1.2 *p-group isomorphism is polynomial time many-one reducible to valence $p+1$ depth $O(\log n)$ cone graph isomorphism.*

Theorem 1.3 *Group Isomorphism for p-groups with n elements given in table representation is in time (for a constant c):* n^{cp}

Remark. The graph construction in Theorem 5.1 can be done in AC^0. By Corollary 4.10 a composition series in Algorithm 1 can be computed in SAC^2. We get the following.

Corollary 5.7 *p-group isomorphism is SAC^2 many-one reducible to valence $p+1$ depth $O(\log n)$ cone graph isomorphism.*

6 Isomorphism test for groups

In this section we give three isomorphism tests. The first is the classical isomorphism test, it can be seen as a variant of the algorithm attributed to Tarjan, c.f. [Mil78]. The second algorithm is an extension of what we introduced for p-groups in Section 5. The third is a combination of both algorithms which improves the known upper bounds for group isomorphism.

For all these algorithms we assume, that for a group G a sequence of composition factors is given with respect to a composition series S and that for S we have a complete set of coset representatives $\vec{s}$. We address the sequence of composition factors of S by $\mathsf{seq}(S)$. Note, any element in a coset could be taken as a representative. So an isomorphism just mapps cosets blockwise onto each other.

6.1 Classical Isomorphism Test

The isomorphism test. Let G and H be two groups. By Lemma 4.9, a composition series S for G and S' for H can be computed in polynomial time such that if G is isomorphic to H then also $(G, \mathsf{seq}(S))$ is isomorphic to $(H, \mathsf{seq}(S'))$.

Let (H, S') be a group with composition series S' and a complete set of coset representatives $\vec{h} = (h_1, \ldots, h_{k-1})$. For an isomorphism test, we guess a complete set of coset representatives $\vec{g} = (g_1, \ldots, g_{k-1})$ in G which are mapped onto $(h_1, \ldots, h_{k-1})$ in this order. Note, if G_i/G_{i+1} is cyclic, then $g_i = (a_i)$ and if G_i/G_{i+1} is non-abelian and simple, then $g_i = (a_i, b_i)$ has two elements. We assume that $g_i = (a_i, b_i)$ is mapped onto $h_i = (a'_i, b'_i)$ by mapping a_i onto a'_i and b_i onto b'_i.

We write the group elements in generator-representation and arrange them in increasing lexicographical order according to their representation. We relabel the elements according to their new order as in Lemma 4.2. We write the multiplication tables for G and H where the elements are sorted and compare them line by line and bit by bit.

We give some notes to Algorithm 2.

Step 1: *compute a composition series for G and H.* Together with the composition series we have coset representatives, i.e. one or two generators for each factor group. By Lemma 4.9, $(G, \mathsf{seq}(S))$ is isomorphic to $(H, \mathsf{seq}(S'))$ iff G is isomorphic to H. These generators must not be mapped later in this algorithm elementwise onto each other.

Step 2: *guess a complete set of coset representatives for (G, S).* For each factor group we guess one or two generators. When the i-th tuple of generators g_i is mapped onto the i-th tuple of generators h_i then we will accept if this mapping induces an isomorphism of $(G, \mathsf{seq}(S))$ onto $(H, \mathsf{seq}(S'))$.

Step 3: *For each group element g, compute its generator representation.* The representation depends on the generators in $\vec{g}$ and $\vec{h}$. The representation for a group element $g \in G$ is a product of generators with unique words according to Lemma 4.2. For example, if G_i/G_{i+1} is cyclic, then $g_i = (a_i)$ and the i-th term in this product is $a_i^{l_i}$ with $l_i \in \{0, \ldots, \mathsf{ord}(a_i) - 1\}$. Whereas if G_i/G_{i+1} is non-abelian and simple, then $g_i = (a_i, b_i)$ and the i-th term is a word with a_i, b_i.

Algorithm 2 Isomorphism testing for Cayley Groups

Input: multiplication tables G, H of two groups with n elements

Computation: accept if G is isomorphic to H and reject otherwise

```
1: compute $S$ for $G$ and $S'$ for $H$ composition series together with complete sets of coset representatives $(g'_1, \dots, g'_{k-1})$ and $\vec{h} = (h_1, \dots, h_{k-1})$ according to Algorithm 1.
2: guess a complete set of coset representatives $\vec{g} = (g_1, \dots, g_{k-1})$ with respect to $(G, S)$
3: for each $g \in G$ (or $H$) do compute $repr(g)$ a word with $g_1, \dots g_{k-1}$ in $G$ with respect to $\vec{g}$ or a word with $h_1, \dots, h_{k-1}$ in $H$ with respect to $\vec{h}$
4: relabel generators according to their order in $\vec{g}$ (and $\vec{h}$)
5: let $T_G$ be $G$ (and $T_H$ be $H$) in table representation where elements are sorted according to their new labels in increasing lexicographical order
6: compare $T_G, T_H$ lexicographically line by line and bit by bit
7: if $T_G = T_H$ then accept and halt.
8: reject and halt.
```

Step 4: *Relabel generators according to their order in $\vec{g}$ (and $\vec{h}$).* The new labels are taken from the set $\{1, \dots, n\}$ in increasing order.

Step 5: *Compute the multiplication table where elements are sorted by their new labels.* We relabel the group elements as in Lemma 4.2 and sort them in increasing lexicographical order.

Step 6 to 8: *Accept iff the tables T_G and T_H are equal.* The comparison is done lexicographically line by line and for each line element by element. The elements are compared bit by bit. If $T_G = T_H$ then we accept, else we reject.

Complexity. It can be verrified that every step can be done in polynomial time by an NP-machine. We calculate now the demand on non-deterministic bits in Algorithm 2.

In Step 3, we guess generators, these are different to the coset representatives from Step 2. An isomorphism then mapps generators onto each other. For $i \in \{1, \dots, k-1\}$ we guess $a_i(, b_i) \in G_i \setminus G_{i+1}$, i.e. we need at most $2\log(|G_i| - |G_{i+1}|) \leq 2\log n$ bits. If G_i/G_{i+1} has order p, then the number of such factor groups is at most $\log_p n$. For an upper bound, we consider p to be the order of the smallest factor group. Hence, we get $O(\log n) \cdot \log_p n$ non-deterministic bits for all generators. We put this together and get the following Theorem.

Theorem 6.1 *Let p be the order of the smallest factor group in S of G. Group Isomorphism on groups with n elements given in table representation can be tested by an NP-machine with access to at most $O(\log n \log_p n)$ non-deterministic bits.*

Remarks. Instead of non-determinism, the computations can be done in deterministic time $n^c \cdot 2^{\log n \log_p n} = n^{c+\log_p n}$ for a constant c.

In the worst case we have $p = 2$ and then we get the upper bound from Tarjans algorithm. Later, for the complexity analysis of the third isomorphism test it is important to consider this bound to be parameterized with p.

For solvable groups, we just have cyclic factor groups and hence, p is the smallest prime that divides n.

The algorithm in [Wag10] for group isomorphism testing is based on *cube generating sequences* and runs in β_2L, i.e. a logspace machine with access to at most $O(\log^2 n)$ non-deterministic bits given additionally to the input. It can be verrified that Algorithm 2 also runs in $\beta_2\mathsf{L}$, the space requirement depends on parameter p. If p is not a constant, then less than $O(\log^2 n)$ bits are required.

6.2 Isomorphism test: Reduction to Bounded Valence Graph Isomorphism.

In the second isomorphism testing algorithm we fix a composition series and then we reduce the isomorphism problem onto graph isomorphism. The valence of the resulting graph depends on the order of the largest factor group.

The reduction. We use the composition series for groups and generalize the reduction of Theorem 5.1 from p-groups to arbitrary groups.

Let (G, S) be a group G over n elements with a composition series S where each factor group G_i/G_{i+1} has order p_i:

$$\{1\} = G_k \lhd G_{k-1} \lhd \cdots \lhd G_1 = G$$

and let $\vec{s} = (s_1, \ldots, s_{k-1})$ be a complete set of coset representatives for S with $a_iG_{i+1} \in G_i/G_{i+1}$ if a_i is in s_i. Correspondingly, let (H, S') be a group H with a complete set of coset representatives $\vec{s'}(s'_1, \ldots, s'_{k-1})$ and composition series S':

$$\{1\} = H_k \lhd H_{k-1} \lhd \cdots \lhd H_1 = H$$

If the factor groups do not have the same order, or are not of the same type, then (G, S) is not isomorphic to (H, S'). We prove the following theorem.

Theorem 6.2 *Let $p = \max\{p_1, \ldots, p_k\}$. There is an AC^0-computable function that computes a graph $X(G, S)$ and $X(H, S')$ with at most $11n^2 + 1$ vertices which have valence at most $p + 1$ such that $X(G, S)$ is isomorphic to $X(H, S')$ if and only if $(G, \mathsf{seq}(S))$ is isomorphic to $(H, \mathsf{seq}(S'))$.*

Proof. First, we generalize the construction of $X_p(G, S)$ in the proof of Theorem 5.1 from p-groups to groups and define a new graph $X(G, S)$.

First, we construct a tree $T(G, S)$ the same way as $T_p(G, S)$ in the proof of Theorem 5.1. The main difference is, that the nodes in the tree have at the i-th level p_i children, i.e. when G_i/G_{i+1} has order p_i. For an example see Figure 4. If one of the factor groups G_i/G_{i+1} is non-abelian and simple, e.g. A_5 the alternating group on 5 elements, then we have $p_i = |G_i/G_{i+1}|$. This number is no longer a prime, e.g. $|A_5| = 60$.

The rest of the construction of $X(G, S)$ is identical to the construction of $X_p(G, S)$ in the proof of Theorem 5.1.

We prove now that $X(G, S)$ has all the properties stated in Theorem 6.2.

Claim 6.3 *The graph $X(G, S)$ has at most $11n^2 + 1$ vertices.*

The proof goes the same lines as in the proof of Claim 5.2. In $X(G, S)$, the inner nodes do not have the same valency, but every node has at least two children. Hence, we get at most $11n^2 + 1$ vertices.

Claim 6.4 *There is a logspace-computable function that computes the graph $X(G, S)$.*

The proof goes the same lines as the proof of Claim 5.3.

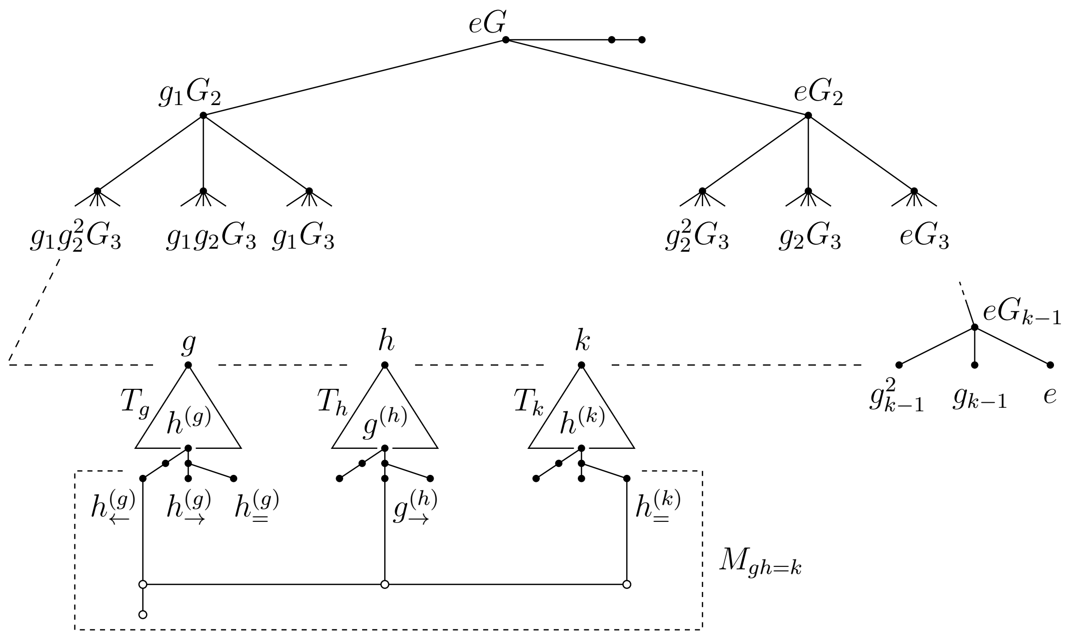

Figure 4: The graph $X(G, S)$ with $p_1 = 2$, $p_2 = 3$ and $p_3 = 5$ is shown together with a multiplication graph gadget $M_{gh=k}$ indicated by white nodes, it is connected to element vertices g, h, k.

Claim 6.5 *The graph $X(G, S)$ has valence at most $p + 1$ for any $p \geq 2$.*

Recall, that there is no factor group of order greater than p. The proof follows the lines of the proof of Claim 5.4.

Claim 6.6 *The graph $X(G, S)$ is isomorphic to $X(H, S')$ if and only if $(G, \mathsf{seq}(S))$ is isomorphic to $(H, \mathsf{seq}(S'))$.*

The proof is similar to the proof of Claim 5.5. In the proof we have a different tree structure, namely $T(G, S)$ instead of $T_p(G, S)$. Both trees are rooted and complete. In $T(G, S)$ nodes at the same distance to the root have the same valence. Hence, there are automorphisms that map a vertex onto every other vertex which has the same distance to the root node eG_1.

This completes the proof of Theorem 6.2. $\square$

Complexity. By Lemma 4.9 we compute S for G and S' for H in polynomial time, such that if $G \cong H$ then also $(G, \mathsf{seq}(S)) \cong (H, \mathsf{seq}(S'))$.

The isomorphism test runs for a constant c in time n^c plus the time complexity for bounded valence GI $n^{c(p+1)}$ (see Theorem 5.6, [BL83]). We get the following theorem.

Theorem 6.7 *Let p be the length of the composition factor with largest size among those in S and S'. Group Isomorphism on groups with n elements given in table representation can be tested in deterministic time $O(n^c + n^{c\cdot(p+1)})$ for a constant c.*

Remarks. This isomorphism test itself is useful for groups with small composition factors. For example, if the group is solvable and n is a product of small primes. Then we see, that this algorithm runs for such instances fast whereas the first algorithm reaches its worst-case runtime. Hence, this motivates the question of a combination of both algorithms to improve the total runtime for group isomorphism.

6.3 Combine both Isomorphism Tests

The complexity of Algorithm 2 depends on the smallest order of factor groups, whereas the complexity of the second algorithm depends on the largest order of factor groups.

The idea is that we use both algorithms as a subroutine in the new algorithm. The algorithm gets additionally to the input a number α as parameter. It has two parts. For factor groups of size larger than α we guess the generators as in Algorithm 2. We modify the construction of the graph $X(G, S)$, such that is has valence at most $\alpha + 1$. Since the number of corresponding factor groups is small, this also keeps the complexity of Algorithm 2 small in the new algorithm.

With parameter α we will improve the runtime of the new isomorphism testing algorithm.

Changes to the graphs. Since we said that we guess some of the generators, we make changes to the graphs from the reduction. We modify $X(G, S)$ for a group G and a composition series S that is given by a complete set of coset representatives $\vec{s} = (s_1, \dots, s_{k-1})$.

Lemma 6.8 *Let A be a set of generators, a subset of the generators in $s_1, \dots, s_{k-1}$. Let α be the factor group with largest size among those in S which do not have generators in A. Then we define a graph $X(G, S, A)$ of valence at most $\alpha + 1$ that behaves like $X(G, S)$ but where any automorphism fixes the generators in A elementwise.*

Proof. Let $a_i \in A$ be a generator, such that $a_i G_{i+1}$ is a coset representative for a cyclic factor group G_i/G_{i+1} in (G, S). Let $a_i, b_i \in A$ be generators with respect to a non-abelian simple factor group G_i/G_{i+1} in (G, S). Let D_i be the set of nodes at distance i to the root eG. That is, $gG_i \in D_i$ with g is any product of generators in $w_1, \dots, w_i$.

We do the following changes for every node $gG_i \in D_i$ in the graph $X(G, S)$:

- *Remove edges to the children of gG_i.* For example, in the cyclic case remove $\{gG_i, gg_{i+1}^{l_{i+1}} G_{i+1}\}$ where g_{i+1} is a fixed coset representative. In the non-abelian simple case, remove gw_jG_i where w_j is a word with a_i, b_i as in the proof of Lemma 4.2.

- *Arrange edges according to generators a_i (or a_i, b_i).* Let $l = \mathsf{ord}(a_i) - 1$ (or l the order of the group generated by a_i, b_i). Arrange the children from right to left:
$$(ga_i^l G_{i+1}, ga_i^{l-1} G_{i+1}, \dots, ga_i^1 G_{i+1}, gG_{i+1}).$$
This is done according to Lemma 4.2.

- *Connect the children from right to left to the leafs of a binary tree with root gG_i such that children have the same distance to gG_i.* We connect pairwise leafs or subtrees from right to left to form larger subtrees. The tree contains nodes with one or two children. See Figure 5 for an example.

- *Color the leaf connected to ga_iG_{i+1} (or the leafs connected to ga_iG_{i+1} and gb_iG_{i+1}).* In the cyclic case, we connect ga_iG_{i+1} to an extra vertex. In the non-abelian simple case, we connect ga_iG_{i+1} to an extra vertex and gb_iG_{i+1} to a path of length three. These can be distinguished from all other vertices, because there is no vertex with the same distance to the root eG connected to a single vertex or a path of length three (we ignore nodes that come from other paths which serve as a coloring of nodes).

The colored nodes ga_iG_{i+1} (or ga_iG_{i+1} and gb_iG_{i+1}) are fixed. If a_iG_{i+1} (or a_iG_{i+1} and b_iG_{i+1}) are fixed in G_i/G_{i+1}, then every automorphism of the group fixes all cosets of the factor group G_i/G_{i+1}. Hence, all children of the node gG_i are fixed in any automorphism of the group.

The tree structure guarantees that the distances to the root eG remain unchanged for all vertices. The graph $X(G,S)$ is a cone graph, it follows that this also holds for $X(G,S,A)$.

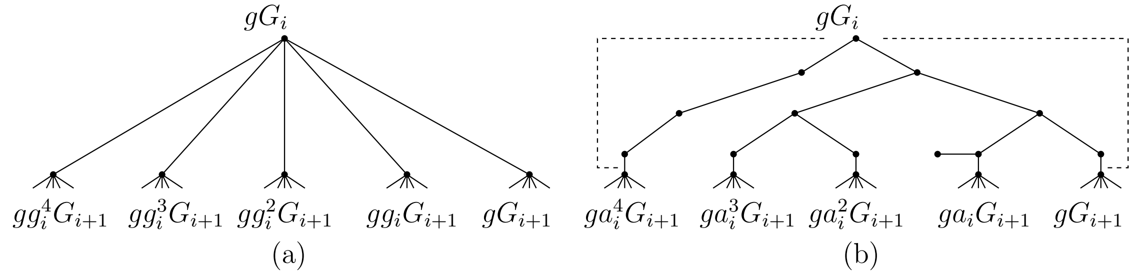

Figure 5: The situation is shown where G_i/G_{i+1} is a cyclic factor group.
(a) The node gG_i and its children $gg_i^jG_{i+1}$ for all $j \in \{0,\dots,l\}$ in $X(G,S)$, with $l=4$.
(b) The graph gadget connected to these nodes in $X(G,S,A)$ enclosed within the dashed box. Note, a_i can be any element in $G_i \setminus G_{i+1}$.

□

Algorithm 3 Isomorphism testing for Cayley Groups
Input: multiplication tables G,H of two groups with n elements, parameter α
Computation: accept if G is isomorphic to H and reject otherwise

```
1: let $S$ be $\vec{s} = (s_1,\dots,s_{k-1}) = \mathsf{compSeries}(G)$ a composition series for $G$ as in Lemma 4.9.
2: let $S'$ be $\vec{s'} = (s'_1,\dots,s'_{k-1}) = \mathsf{compSeries}(H)$ a composition series for $H$, accordingly.
3: for $i \in \{1,\dots,k-1\}$ do
4:     if $G_i/G_{i+1}$ is of order $> \alpha$ then
5:         guess $t = (a)$ (or $t = (a,b)$) with $\langle aG_{i+1}, bG_{i+1}\rangle \cong G_i/G_{i+1}$
6:         $t_i = t$, $A \leftarrow a,b$, $A' \leftarrow a',b' \in s'_i$
7:     end if
8: end for
9: if $X(G,S,A) \cong X(H,S',A')$ then accept and halt
10: reject and halt
```

The algorithm. We give some notes to Algorithm 3. In Lines 1 and 2, we compute a composition series S for G and S' for H together with coset representatives that we obtain from Algorithm 1.

In Lines 3 and 4, we run through the factor groups of order greater than α.

In Line 5, for each such factor group G_i/G_{i+1} we guess one or two generators t_i, depending on whether the factor group is cyclic or non-abelian and simple.

In Line 6, we put the guessed generators in t_i also in a set A and correspondingly those in s'_i in a set A'.

In Line 9, we construct the graphs where we treat generators in the sets A and A' specially. If the graphs are isomorphic, then we accept and halt.

In Line 10, we reject and halt.

The complexity of group isomorphism. We calculate the runtime of Algorithm 3.
In Lines 1 and 2, the composition series can be computed in polynomial time according to Lemma 4.9.
In Line 5, we say that we guess generators for composition factors of size $> \alpha$. For this we need $\log n \log_\alpha n$ non-deterministic bits only. In other words, Algorithm 3 runs several times, trying all possibilities as generators. Hence, the running time is multiplied with $2^{\log n \log_\alpha n} = n^{\log_\alpha n}$.
In Line 9, we invoke an isomorphism testing algorithm for graphs of valence at most $\alpha + 1$. This algorithm runs in time $n^{O(\alpha+1)}$.

Hence, we get as total running time (for a constant c):

$$n^c + n^{c \log_\alpha n} + n^{c\alpha}$$

This becomes minimal if $\log_\alpha n = \alpha$, that is

$$\log n / \log \alpha = \alpha$$

Now, we substitute α by $\log n/\log\alpha$ on the left side and get

$$\begin{aligned} \log n / \log(\log n / \log \alpha) &= \alpha & (1) \\ \log n / (\log\log n - \log\log\alpha) &= \alpha & (2) \end{aligned}$$

If we substitute α again by $\log n/\log\alpha$ on the left side, and with $\log\log\alpha \leq \log\log\log n$ we proved the main theorem.

Theorem 1.1 *Group isomorphism for groups with n elements given in table representation is in time (for a constant c):*

$$n^{c \log n / \log\log n}$$

Conclusion

We improve the complexity of group isomorphism when groups of order n are given in table representation. We proved that isomorphism testing for p-groups is in polynomial time if p is a constant prime. Hence, the worst case moves from $p = 2$ close to $\log n$. In the case of solvable groups, the complexity depends heavily on the factorization of n. For groups in general, we improve the known upper bound from $n^{\log n + O(1)}$ to $n^{c \log n / \log\log n}$.

It would be interesting whether the new techniques can be adapted to isomorphism testing for quasigroups.

Acknowledgements. We thank Jacobo Torán for discussions and comments on the manuscript.

References

[AT04] V. Arvind and Jacobo Torán. Solvable group isomorphism is (almost) in NP $\cap$ coNP. In *Annual IEEE Conference on Computational Complexity (formerly Annual Conference on Structure in Complexity Theory)*, volume 19, 2004.

[AV04] V. Arvind and T. C. Vijayaraghavan. Abelian permutation group problems and logspace counting classes. In *Annual IEEE Conference on Computational Complexity (formerly Annual Conference on Structure in Complexity Theory)*, volume 19, pages 204–214, 2004.

[BCGQ11] László Babai, Paolo Codenotti, Joshua A. Grochow, and Youming Qiao. Code equivalence and group isomorphism. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1395–1408, 2011.

[BL83] László Babai and Eugene M. Luks. Canonical labeling of graphs. In *15th Annual ACM Symposium on Theory of Computing (STOC)*, pages 171–183, 1983.

[CTW10] Arkadev Chattopadhyay, Jacobo Torán, and Fabian Wagner. Graph isomorphism is not AC^0 reducible to group isomorphism. In *Proceedings of the 30th Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 2010.

[DLN08] Samir Datta, Nutan Limaye, and Prajakta Nimbhorkar. 3-connected planar graph isomorphism is in log-space. In *Proceedings of the 28th annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 153–162, 2008.

[Faw09] Joanna Fawcett. The o'nan-scott theorem for finite primitive permutation groups, and finite representability. Thesis, University of Waterloo, UWSpace *http://hdl.handle.net/10012/4534*, 2009.

[Hal99] Marshall Hall. *The theory of groups.* AMS Chelsea Publishing, American Mathematical Society, Providence, Rhode Island, 1999.

[Kav03] Telikepalli Kavitha. Efficient algorithms for abelian group isomorphism and related problems. *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 23, 2003.

[LSZ76] Richard J. Lipton, Lawrence Snyder, and Yechezkel Zalcstein. The complexity of word and isomorphism problems for finite groups. Technical report, John Hopkins, 1976.

[Luk93] Eugene M. Luks. Permutation groups and polynomial-time computation. *DIMACS series in Discrete Mathematics and Theoretical Computer Science*, 11:139–175, 1993.

[Mil78] Gary L. Miller. On the $n^{log n}$ isomorphism technique. In *ACM Symposium on Theory of Computing (STOC)*, 1978.

[Mil79] Gary L. Miller. Graph isomorphism, general remarks. *Journal of Computer and System Sciences*, 18(2):128–142, 1979.

[MSW94] Gunter Malle, Jan Saxl, and Thomas Weigel. Generation of classical groups. *Geometriae Dedicata*, 49(1):85–116, 1994.

[Pap94] Christos M. Papadimitriou. *Computational complexity.* Addison-Wesley, Reading, Massachusetts, 1994.

[QMT11] Youming Qiao, Jayalal Sarma M.N., and Bangsheng Tang. On isomorphism testing of groups with normal hall subgroups. In *28st Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, 2011.

[Rei08] Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM (JACM)*, 55(4):1–24, 2008.

[Wag10] Fabian Wagner. *Isomorphism Testing for Restricted Graph Classes - On the complexity of isomorphism testing and reachability problems for restricted graph classes.* Süddeutscher Verlag für Hochschulschriften, 2010.