Hypergraph Isomorphism for Groups with Restricted Composition Factors

Daniel Neuen
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
dneuen@mpi-inf.mpg.de

Abstract

We consider the isomorphism problem for hypergraphs taking as input two hypergraphs over the same set of vertices V and a permutation group Γ over domain V, and asking whether there is a permutation $\gamma \in \Gamma$ that proves the two hypergraphs to be isomorphic. We show that for input groups, all of whose composition factors are isomorphic to a subgroup of the symmetric group on d points, this problem can be solved in time $(n+m)^{\mathcal{O}((\log d)^c)}$ for some absolute constant c where n denotes the number of vertices and m the number of hyperedges. In particular, this gives the currently fastest isomorphism test for hypergraphs in general. The previous best algorithm for the above problem due to Schweitzer and Wiebking (STOC 2019) runs in time $n^{\mathcal{O}(d)}m^{\mathcal{O}(1)}$.

As an application of this result, we obtain, for example, an algorithm testing isomorphism of graphs excluding $K_{3,h}$ as a minor in time $n^{\mathcal{O}((\log h)^c)}$. In particular, this gives an isomorphism test for graphs of Euler genus at most g running in time $n^{\mathcal{O}((\log g)^c)}$.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Mathematics of computing → Graph algorithms; Mathematics of computing → Combinatorial algorithms; Mathematics of computing → Graphs and surfaces

Keywords and phrases graph isomorphism, groups with restricted composition factors, hypergraphs, bounded genus graphs

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.88

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/2002.06997`.

1 Introduction

Luks's algorithm [21] is an important cornerstone of the algorithmic theory of the Graph Isomorphism Problem. With some additional improvements given later [6], it tests in time $n^{\mathcal{O}(d/\log d)}$ whether two given n-vertex graphs of maximum degree d are isomorphic. In the last four decades, Luks's algorithm has developed to a central building block for many algorithms tackling the isomorphism problem. Indeed, Luks's algorithmic framework has been used as a subroutine to design, for example, isomorphism tests for graphs of bounded genus [27], graphs of bounded tree-width [15], graphs excluding some fixed graph as a minor [31], and even graph classes that exclude some fixed graph as a topological minor [13]. Further examples include color-t-bounded graphs [4], defined by Babai et al. in the context of isomorphism testing of strongly regular graphs, unit square graphs [28], and graphs of bounded rank-width [17]. Moreover, Luks's algorithm has also played a role in the area of computational group theory for example for computing normalizers for certain groups [23].

Additionally, Luks's algorithm also forms the basis for Babai's recent quasipolynomial isomorphism test [1] as well as the corresponding canonization algorithm [2]. Indeed, Babai's algorithm follows the recursive framework of Luks's algorithm and attacks the obstacle cases where the recursion performed by Luks's algorithm does not lead to the desired running time. Hence, a natural question to ask is whether it is possible extend the group-theoretic methods added in Babai's algorithm to the setting of bounded degree graphs in order to obtain improved algorithms also for the isomorphism problem for graphs of small degree. This question was answered in the affirmative by Grohe, Schweitzer and the author of this paper in [14] providing an isomorphism test for graphs of maximum degree d running in time $n^{\mathcal{O}((\log d)^c)}$ for some constant c.

With the large number of applications of Luks's algorithmic framework [21] over the last decades it is natural to ask for improvements of other algorithms that exploit Luks's methods as a subroutine. However, up to this point, the only application of the improved isomorphism test for graphs of small degree is a faster fixed-parameter tractable isomorphism test for graphs of bounded tree-width [15]. The reason for this mainly lies in the fact that most of the algorithms exploiting Luks's framework as a subroutine actually use this framework to solve more general problems than Luks's original algorithm.

To be more precise, Luks's original algorithm [21] attacks the *String Isomorphism Problem*. The input to the String Isomorphism Problem are two strings $\mathfrak{x}, \mathfrak{y} \colon \Omega \to \Sigma$, where Ω is a finite set and Σ a finite alphabet, and a permutation group $\Gamma \leq \mathrm{Sym}(\Omega)$ (given by a set of generators). The task of the String Isomorphism Problem is to decide whether there exists some $\gamma \in \Gamma$ that transforms $\mathfrak{x}$ into $\mathfrak{y}$. In order to solve the isomorphism problem for graphs of bounded degree, Luks [21] provides a polynomial-time algorithm solving the String Isomorphism Problem for all input groups Γ in the class $\widehat{\Gamma}_d$[1], the class of groups all of whose composition factors are isomorphic to a subgroup of S_d (the symmetric group on d points). To give a faster isomorphism test for graph of small degree, Grohe et al. [14] follow the same route of considering the String Isomorphism Problem and provide an algorithm solving the problem in time $n^{\mathcal{O}((\log d)^c)}$ for the class of $\widehat{\Gamma}_d$-groups.

On the other hand, many algorithms exploiting the methods of Luks do so by solving more general problems. Indeed, a common extension is the Hypergraph Isomorphism Problem for $\widehat{\Gamma}_d$-groups. Here, the input consists of two hypergraphs $\mathcal{H}_1 = (V, \mathcal{E}_1)$ and $\mathcal{H}_2 = (V, \mathcal{E}_2)$ over the same set of vertices and a $\widehat{\Gamma}_d$-group $\Gamma \leq \mathrm{Sym}(V)$, and the task is to decide whether there is some $\gamma \in \Gamma$ that transforms $\mathcal{H}_1$ into $\mathcal{H}_2$. As observed by Miller [26], with some small modifications, Luks's algorithm for the String Isomorphism Problem immediately extends to the Hypergraph Isomorphism Problem. To be more precise, by an extension of the arguments of Luks [21], the Hypergraph Isomorphism Problem for $\widehat{\Gamma}_d$-groups can be solved in time $(m+n)^{\mathcal{O}(d)}$ where n denotes the number of vertices and m the number of edges. This fact is exploited by several of the algorithms mentioned above. For example, this includes the isomorphism tests for graphs of bounded genus [27], graphs excluding some fixed graph as a (topological) minor [31], color-t-bounded graphs [4], and unit square graphs [28]. Recently, an improved algorithm for the Hypergraph Isomorphism Problem for $\widehat{\Gamma}_d$-groups running in time $n^{\mathcal{O}(d)}m^{\mathcal{O}(1)}$ was presented by Schweitzer and Wiebking [34]. While the algorithm of Schweitzer and Wiebking significantly improves the running for large numbers of hyperedges, this is irrelevant for the applications described above where the number of hyperedges is typically linearly bounded in the number of vertices of the original input graph.

The main contribution of this paper is to provide an algorithm for the Hypergraph Isomorphism Problem for $\widehat{\Gamma}_d$-groups running in time $(n+m)^{\mathcal{O}((\log d)^c)}$ for some absolute constant c. Besides its potential applications outlined above, the Hypergraph Isomorphism Problem is of course also interesting in its own right. Indeed, there is also a long history of studying the Hypergraph Isomorphism Problem in itself. One of the first results in this direction was an algorithm testing isomorphism of hypergraphs in time $2^{\mathcal{O}(n)}$ [22] where

[1] In [21] this class is denoted by Γ_d. However, in the more recent literature, Γ_d often refers to a more general class of groups.

n denotes the number of vertices. Assuming the hyperedges are not too large, this result was improved by Babai and Codenotti in [5] to a running of $n^{\widetilde{\mathcal{O}}(k^2 \cdot \sqrt{n})}$ where $\widetilde{\mathcal{O}}(\cdot)$ hides polylogarithmic factors and k denotes the maximal size of a hyperedge. Again assuming small hyperedges, another improvement follows from the work of Grohe et al. [14] resulting in an isomorphism test for hypergraphs running in time $n^{\mathcal{O}(k \cdot (\log n)^c)}$ for a constant c.

Results. The main result of this paper is a faster algorithm for the Hypergraph Isomorphism Problem for $\widehat{\Gamma}_d$-groups.

▶ **Theorem 1.** *The Hypergraph Isomorphism Problem for $\widehat{\Gamma}_d$-groups can be solved in time $(n+m)^{\mathcal{O}((\log d)^c)}$ for some absolute constant c where n denotes the number of vertices and m the number of edges of the input hypergraphs.*

An immediate consequence of this result is the fastest algorithm for testing isomorphism of hypergraphs in general (assuming the number of hyperedges is moderately exponentially bounded in the number of vertices, i.e., $m = 2^{\mathcal{O}(n^{1-\varepsilon})}$ for some $\varepsilon > 0$).

▶ **Corollary 2.** *The Hypergraph Isomorphism Problem can be solved in time $(n+m)^{\mathcal{O}((\log n)^c)}$ for some absolute constant c where n denotes the number of vertices and m the number of edges of the input hypergraphs.*

In particular, this result removes the dependence on k, the maximal size of a hyperedge, in the running time. Observe that this improvement is significant if k is large and m is small compared to $\binom{n}{k}$ (which is typical for many applications). Also, the last theorem generalizes the corresponding result for the String Isomorphism Problem for $\widehat{\Gamma}_d$-groups obtained in [14].

For the algorithm, we take a similar route than the algorithm from [14] first normalizing the input to ensure a suitable variant of Babai's Unaffected Stabilizers Theorem [1]. Based on this variant, Grohe et al. are able to extend the Local Certificates Routine, a key subroutine of Babai's quasipolynomial-time algorithm [1], to the setting of $\widehat{\Gamma}_d$-groups. Unfortunately, when going from strings to hypergraphs, there is no simple extension of the Local Certificates Routine even in the normalized setting from [14]. Intuitively speaking, the reason is that the Local Certificates Routine crucially exploits that positions in disjoint sets can be permuted independently of each other which is not the case for hypergraphs.

To circumvent this problem we introduce a novel simplification routine which is repeatedly executed during the Local Certificates Routine. The idea for the simplification is based on the following observation. Suppose that all hyperedges are identical on a window $W \subseteq V$ (i.e., $E_1 \cap W = E_2 \cap W$ for all hyperedges $E_1, E_2 \in \mathcal{E}$). In this case each permutation in $\Gamma_{(V \setminus W)}$ (the subgroup that fixes each position outside of W) is an automorphism as required by the Local Certificates Routine since there are no additional dependencies between W and $V \setminus W$ coming from the hyperedges. The main idea is to always reduce to this simple case. Intuitively speaking, this is achieved as follows. Two hyperedges $E_1, E_2 \subseteq V$ are *W-equivalent* if $E_1 \cap W = E_2 \cap W$. The algorithm creates, for each equivalence class, a copy of the vertex set and associates all hyperedges from the equivalence class with this copy. After this modification, each copy satisfies the requirement that all hyperedges are W-equivalent (actually, to realize this modification, we shall consider a more general problem). This enables us to implement a Local Certificates Routine for hypergraphs which builds the central subroutine of our isomorphism test. Unfortunately, while this settles the original problem, it also creates several additional issues that need to be addressed and which require various extensions of existing techniques making the entire algorithm quite complicated.

With the Hypergraph Isomorphism Problem for $\widehat{\Gamma}_d$-groups being used as a subroutine in a number of algorithms, it is natural to ask for the implications of this result. One of the first examples in this direction is Miller's algorithm for testing isomorphism of graphs of bounded genus [27]. However, Miller's algorithm reduces the isomorphism problem for graphs of genus g to the Hypergraph Isomorphism Problem where the input group is a Γ_d*-tower* rather than a $\widehat{\Gamma}_d$-group where $d = \mathcal{O}(g)$. The class of Γ_d-towers extends the class $\widehat{\Gamma}_d$ by, intuitively speaking, allowing to combine groups that are "almost" $\widehat{\Gamma}_d$-groups along a specified partition of the domain.

Since it is already completely unclear how to extend the algorithm of Grohe et al. [14] to Γ_d-towers it is not possible to use Miller's algorithm directly. To still obtain a faster isomorphism test for graphs of small genus, we strengthen Miller's result and develop a reduction from the isomorphism problem for graphs of genus g to the Hypergraph Isomorphism Problem for $\widehat{\Gamma}_d$-groups where $d \coloneqq 4g + 2$. Actually, our reduction works for any graph class that excludes $K_{3,h}$ as a minor (graphs of genus g exclude $K_{3,4g+3}$ as a minor [32, 18]).

To build the reduction, we introduce *t-CR-bounded graphs* which extend the notion of color-t-bounded graphs introduced in [4]. Intuitively speaking, a vertex-colored graph is t-CR-bounded if the vertex-coloring of the graph can be turned into a discrete coloring (i.e., each vertex has its own color) by repeatedly applying the standard Color Refinement algorithm (see, e.g., [10, 19]) and by splitting color classes of size at most t. We show that the isomorphism problem for t-CR-bounded graphs can be solved in time $n^{\mathcal{O}((\log t)^c)}$ by providing a reduction to the Hypergraph Isomorphism Problem for $\widehat{\Gamma}_t$-groups. While this result may already be interesting in its own right, the main purpose of the isomorphism problem for t-CR-bounded graphs in this paper is to serve as an intermediate problem.

We continue to prove that the isomorphism problem for graph classes that exclude $K_{3,h}$ as a minor is polynomial-time reducible to testing isomorphism of t-CR-bounded graphs where $t \coloneqq h - 1$. This implies the following theorem.

▶ **Theorem 3.** *The Graph Isomorphism Problem for graphs that exclude $K_{3,h}$ as a minor can be solved in time $n^{\mathcal{O}((\log h)^c)}$ for some absolute constant c where n denotes the number of vertices of the input graphs.*

The previous best algorithm for this setting is due to Ponomarenko [31] who provided a polynomial-time isomorphism test for graph classes that exclude an arbitrary minor. While Ponomarenko does not provide a precise analysis on the running time of his algorithm, the exponent depends at least linearly on h when excluding K_h as a minor. Hence, in the case of excluding $K_{3,h}$ as a minor, the above theorem significantly improves on all previous algorithms.

Finally, exploiting the fact that $K_{3,h}$ has Euler genus linear in h [32, 18], we obtain the following corollary.

▶ **Corollary 4.** *The Graph Isomorphism Problem for graphs of Euler genus at most g can be solved in time $n^{\mathcal{O}((\log g)^c)}$ for some absolute constant c where n denotes the number of vertices of the input graphs.*

While the isomorphism problem for graphs of bounded genus is also fixed-parameter tractable [20] (again, the author does not analyze the dependence of the running time on g), the algorithm from the previous corollary is guaranteed to be faster as soon as the genus passes a threshold that is polylogarithmic in the number of vertices. Also, I remark that the isomorphism problem for graphs of bounded genus can be solved in logarithmic space [12].

As a another application of the isomorphism problem for t-CR-bounded graphs, we consider the isomorphism problem for hereditarily finite sets (see also [34]). Intuitively speaking, a hereditarily finite set over ground set V is any type of combinatorial object that can be built by iteratively taking sets and tuples over previously created objects. In particular, hereditarily finite sets include graphs, vertex- and arc-colored graphs, hypergraphs and relational structures. We further extend our previous result on the Hypergraph Isomorphism Problem for $\widehat{\Gamma}_d$-groups and show that isomorphism for any pair of hereditarily finite sets with a $\widehat{\Gamma}_d$-group can be tested in time $(n+m)^{\mathcal{O}((\log d)^c)}$ where n denotes the size of the ground set and m the size of the hereditarily finite sets. In case m is only moderately exponential in n (i.e., $m = 2^{\mathcal{O}(n^{1-\varepsilon})}$ for some fixed $\varepsilon > 0$) this improves over a previous algorithm of Schweitzer and Wiebking [34] (although it should be noted that Schweitzer and Wiebking actually solve a slightly more general problem).

Related Work. Another extension of Babai's quasipolynomial-time algorithm has been independently proposed by Daniel Wiebking [37] providing another proof that the isomorphism problem for arbitrary hereditarily finite sets can be solved in quasipolynomial time. However, Wiebking not only solves the isomorphism problem for hereditarily finite sets as defined in this paper, but actually provides an algorithm for an extended setting where also implicitly represented labeling cosets are allowed as atomic objects for the hereditarily finite sets. Also, Wiebking gives a canonization algorithm which extends Babai's recent canonization algorithm for graphs [2].

But on the other hand, the algorithmic framework of Wiebking [37] is not able to exploit any restrictions on the input group. Hence, the two results are incomparable with respect to the power of the algorithms obtained.

This is also highlighted by the applications. While the results of this paper allow the design of an algorithm solving the isomorphism problem for graphs of Euler genus at most g running in time $n^{\mathcal{O}((\log g)^c)}$, Wiebking utilizes his algorithm to build an isomorphism test for graphs of tree-width at most k running in time $n^{\mathcal{O}((\log k)^c)}$.

2 Preliminaries

Graphs. A *graph* is a pair $G = (V(G), E(G))$ with vertex set $V(G)$ and edge set $E(G)$. Unless stated otherwise, all graphs are undirected and simple graphs, i.e., there are no loops or multiedges. In this setting an edge is denoted as vw where $v, w \in V(G)$. The *neighborhood* of a vertex v is denoted $N_G(v) := \{w \in V(G) \mid vw \in E(G)\}$. The *degree* of a vertex $v \in V(G)$, denoted $\deg_G(v)$, is the size of its neighborhood. Also, for a set of vertices $X \subseteq V(G)$, the neighborhood of X is defined as $N_G(X) := \left(\bigcup_{v\in X} N_G(v)\right) \setminus X$. Usually, we omit the index G if it is clear from the context and simply write $N(v)$, $N(X)$ and $\deg(v)$. For $X \subseteq V(G)$ the *induced subgraph on* X is $G[X] := (X, \{vw \mid v, w \in X, vw \in E(G)\})$. Also, $G - X := G[V(G) \setminus X]$ denotes the induced subgraph on the complement of X.

Two graphs G and H are isomorphic if there is a bijection $\varphi\colon V(G) \to V(H)$ such that $vw \in E(G)$ if and only if $\varphi(v)\varphi(w) \in E(H)$. In this case φ is an *isomorphism* from G to H, which is also denoted by $\varphi\colon G \cong H$. Moreover, $\mathrm{Iso}(G, H)$ denotes the set of all isomorphisms from G to H. The automorphism group of G is $\mathrm{Aut}(G) := \mathrm{Iso}(G, G)$. Observe that, if $\mathrm{Iso}(G, H) \neq \emptyset$, it holds that $\mathrm{Iso}(G, H) = \mathrm{Aut}(G)\varphi := \{\gamma\varphi \mid \gamma \in \mathrm{Aut}(G)\}$ for every isomorphism $\varphi \in \mathrm{Iso}(G, H)$. The Graph Isomorphism Problem takes as input two graphs G and H and asks whether they are isomorphic.

A *(vertex-)colored graph* is a tuple $G = (V(G), E(G), \chi_V)$ where $\chi_V : V(G) \to C$ is a mapping and C is some finite set of colors. A *vertex- and arc-colored graph* is a tuple $G = (V(G), E(G), \chi_V, \chi_E)$ where $\chi_V : V(G) \to C$ is a vertex-coloring and $\chi_E : \{(v, w) \mid vw \in E(G)\} \to C$ is an arc-coloring, where C is again some finite set of colors. Note that an uncolored graph can be interpreted as a vertex- and arc-colored graph where each vertex is assigned the same color as well as each (directed) edge is assigned the same color. The *vertex color classes* of a (colored) graph are the sets $\chi_V^{-1}(c)$ where $c \in C$. A vertex-coloring χ_V is *discrete* if all color classes are singletons, i.e., $\chi_V(v) \neq \chi_V(w)$ for all distinct $v, w \in V(G)$.

Permutation Groups. Next, we establish the basic notation for permutation groups required for this work. For a general background on group theory I refer to [33] whereas background on permutation groups can be found in [11].

A *permutation group* acting on a set Ω is a subgroup $\Gamma \leq \mathrm{Sym}(\Omega)$ of the symmetric group. The size of the permutation domain Ω is called the *degree* of Γ and, throughout this work, is denoted by $n = |\Omega|$. If $\Omega = [n]$ then we also write S_n instead of $\mathrm{Sym}(\Omega)$. Also, $\mathrm{Alt}(\Omega)$ denotes the alternating group on the set Ω and, similar to the symmetric group, we write A_n instead of $\mathrm{Alt}(\Omega)$ if $\Omega = [n]$. For $\gamma \in \Gamma$ and $\alpha \in \Omega$ we denote by α^γ the image of α under the permutation γ. The set $\alpha^\Gamma := \{\alpha^\gamma \mid \gamma \in \Gamma\}$ is the *orbit* of α. The group Γ is *transitive* if $\alpha^\Gamma = \Omega$ for some (and therefore every) $\alpha \in \Omega$.

For $\alpha \in \Omega$ the group $\Gamma_\alpha := \{\gamma \in \Gamma \mid \alpha^\gamma = \alpha\} \leq \Gamma$ is the *stabilizer* of α in Γ. The group Γ is *semi-regular* if $\Gamma_\alpha = \{\mathrm{id}\}$ for all $\alpha \in \Omega$ (id denotes the identity element of the group). For $A \subseteq \Omega$ and $\gamma \in \Gamma$ let $A^\gamma := \{\alpha^\gamma \mid \alpha \in A\}$. The *pointwise stabilizer* of A is the subgroup $\Gamma_{(A)} := \{\gamma \in \Gamma \mid \forall \alpha \in A : \alpha^\gamma = \alpha\}$. The *setwise stabilizer* of A is the subgroup $\Gamma_A := \{\gamma \in \Gamma \mid A^\gamma = A\}$. For a Γ-invariant set $A \subseteq \Omega$ we denote by $\Gamma[A]$ the induced natural action of Γ on A (i.e., restricting every permutation to the set A).

In order to perform computational tasks for permutation groups efficiently it is infeasible to list all elements of a permutation group. Instead, permutation groups are represented by generating sets of small size. A set $S \subseteq \Gamma$, where $\Gamma \leq \mathrm{Sym}(\Omega)$, is a *generating set* for Γ if every $\gamma \in \Gamma$ can be written as a product $\gamma = s_1 s_2 \dots s_k$ of elements $s_1, \dots, s_k \in S$. Indeed, most algorithms for permutation groups are based on so-called *strong generating sets*, which can be chosen of size quadratic in the degree of the group and can be computed in polynomial time given an arbitrary generating set (see, e.g., [35]). This enables the design of efficient algorithms for many basic computational problems for permutation groups (e.g., membership tests, computing the order of a group, the orbits of a group, and generating sets for pointwise stabilizers). For detailed background on algorithms for permutation groups I refer to [35].

Groups with Restricted Composition Factors. In this work we shall be interested in a particular subclass of permutation groups, namely groups with restricted composition factors. Let Γ be a group. A *subnormal series* is a sequence of subgroups $\Gamma = \Gamma_0 \trianglerighteq \Gamma_1 \trianglerighteq \dots \trianglerighteq \Gamma_k = \{\mathrm{id}\}$. The length of the series is k and the groups Γ_{i-1}/Γ_i are the factor groups of the series, $i \in [k]$. A *composition series* is a strictly decreasing subnormal series of maximal length. For every finite group Γ all composition series have the same family of factor groups considered as a multi-set (cf. [33]). A *composition factor* of a finite group Γ is a factor group of a composition series of Γ.

▶ **Definition 5.** *For $d \geq 2$ let $\widehat{\Gamma}_d$ denote the class of all groups Γ for which every composition factor of Γ is isomorphic to a subgroup of S_d.*

We want to stress the fact there are two similar classes of groups both typically denoted by Γ_d in the literature. One is the class we define as $\widehat{\Gamma}_d$ introduced by Luks [21] while the other one used in [3] in particular allows simple groups of Lie type of bounded dimension as composition factors.

3 Isomorphism for Hypergraphs

In the following we provide a very high-level sketch of the algorithm solving the Hypergraph Isomorphism Problem for $\widehat{\Gamma}_d$-groups. For all details I refer to the full version of this paper [30].

3.1 The Generalized String Isomorphism Problem

For the purpose of building a recursive algorithm, we consider a slightly different problem that crucially allows us to modify instances in a certain way exploited later on.

Let $\Gamma \leq \mathrm{Sym}(\Omega)$ be a group and let $\mathfrak{P}$ be a partition of the set Ω. The partition $\mathfrak{P}$ is Γ-*invariant* if $\mathfrak{P}^\gamma = \mathfrak{P}$ for all $\gamma \in \Gamma$ where $\mathfrak{P}^\gamma := \{P^\gamma \mid P \in \mathfrak{P}\}$. A $\mathfrak{P}$-*string* is a pair $(P, \mathfrak{x})$ where $P \in \mathfrak{P}$ and $\mathfrak{x}\colon P \to \Sigma$ is a string over a finite alphabet Σ. For $\sigma \in \mathrm{Sym}(\Omega)$ the string $\mathfrak{x}^\sigma$ is defined by $\mathfrak{x}^\sigma\colon P^\sigma \to \Sigma\colon \alpha \mapsto \mathfrak{x}(\alpha^{\sigma^{-1}})$. A permutation $\sigma \in \mathrm{Sym}(\Omega)$ is a Γ-*isomorphism* from $(P, \mathfrak{x})$ to a second $\mathfrak{P}$-string $(Q, \mathfrak{y})$ if $\sigma \in \Gamma$ and $(P^\sigma, \mathfrak{x}^\sigma) = (Q, \mathfrak{y})$.

▶ **Definition 6.** *The* Generalized String Isomorphism Problem *takes as input a permutation group* $\Gamma \leq \mathrm{Sym}(\Omega)$*, a* Γ*-invariant partition* $\mathfrak{P}$ *of the set* Ω*, and* $\mathfrak{P}$*-strings* $(P_1, \mathfrak{x}_1), \dots, (P_m, \mathfrak{x}_m)$ *and* $(Q_1, \mathfrak{y}_1), \dots, (Q_m, \mathfrak{y}_m)$*, and asks whether there is some* $\gamma \in \Gamma$ *such that*

$$\{(P_1^\gamma, \mathfrak{x}_1^\gamma), \dots, (P_m^\gamma, \mathfrak{x}_m^\gamma)\} = \{(Q_1, \mathfrak{y}_1), \dots, (Q_m, \mathfrak{y}_m)\}.$$

We usually denote $\mathfrak{X} = \{(P_1, \mathfrak{x}_1), \dots, (P_m, \mathfrak{x}_m)\}$ and $\mathfrak{Y} = \{(Q_1, \mathfrak{y}_1), \dots, (Q_m, \mathfrak{y}_m)\}$. Also, $\mathrm{Iso}_\Gamma(\mathfrak{X}, \mathfrak{Y})$ denotes the set of Γ-isomorphisms from $\mathfrak{X}$ to $\mathfrak{Y}$ and $\mathrm{Aut}_\Gamma(\mathfrak{X}) := \mathrm{Iso}_\Gamma(\mathfrak{X}, \mathfrak{X})$.

It is easy to see that the Hypergraph Isomorphism Problem can be reduced to the Generalized String Isomorphism Problem choosing $\mathfrak{P}$ to be the trivial partition consisting of one block and adding strings $\mathfrak{x}_E\colon V \to \{0,1\}$ for each hyperedge E where $\mathfrak{x}_E(v) = 1$ if and only if $v \in E$.

For the rest of this section we denote by $n := |\Omega|$ the size of the domain, and m denotes the size of $\mathfrak{X}$ and $\mathfrak{Y}$ (we always assume $|\mathfrak{X}| = |\mathfrak{Y}|$, otherwise the problem is trivial). The goal of this section is to provide an algorithm solving the Generalized String Isomorphism Problem for $\widehat{\Gamma}_d$-groups in time $(n+m)^{\mathcal{O}((\log d)^c)}$ for some absolute constant c.

We start by introducing some additional notation. For every $P \in \mathfrak{P}$ define $\mathfrak{X}[[P]] := \{\mathfrak{x}\colon P \to \Sigma \mid (P, \mathfrak{x}) \in \mathfrak{X}\}$ and also let $m_\mathfrak{X}(P) := |\mathfrak{X}[[\mathfrak{P}]]|$. We say that $\mathfrak{X}$ is *completely occupied* if $m_\mathfrak{X}(P) \geq 1$ for every $P \in \mathfrak{P}$. Also, we say that $\mathfrak{X}$ is *simple* if $m_\mathfrak{X}(P) \leq 1$ for every $P \in \mathfrak{P}$.

We will assume throughout this work that all sets of $\mathfrak{P}$-strings encountered are completely occupied, also if not explicitly stated. Note that an instance, which is not completely occupied, can be easily turned into one, that is completely occupied, by introducing additional $\mathfrak{P}$-strings. Also, for a set $A \subseteq \Omega$ and a set of $\mathfrak{P}$-strings $\mathfrak{X}$ define

$$\mathfrak{X}[A] := \{(P \cap A, \mathfrak{x}[A \cap P]) \mid (P, \mathfrak{x}) \in \mathfrak{X}, P \cap A \neq \emptyset\}$$

to be the subinstance induced by A where $\mathfrak{x}[A \cap P]$ denotes the substring induced by $A \cap P$.

3.2 Normalization of the Group Action

The main hurdle to build the algorithm for the Generalized String Isomorphism Problem is an extension of the Local Certificates Routine introduced by Babai for his quasipolynomial time isomorphism test [1]. Similar to [14], our algorithm exploits a variant of the Unaffected Stabilizers Theorem for $\widehat{\Gamma}_d$-group which builds the theoretical foundation for the Local Certificates algorithm.

However, this variant of the Unaffected Stabilizers Theorem for $\widehat{\Gamma}_d$-groups additionally requires the input group to have an *almost d-ary sequence of partitions.*

For a partition $\mathfrak{B}$ of the set Ω and $S \subseteq \Omega$ let $\mathfrak{B}[S] := \{B \cap S \mid B \in \mathfrak{B}, B \cap S \neq \emptyset\}$ denote the induced subpartition of $\mathfrak{B}$ on the set S. Let $\mathfrak{B}'$ be a second partition of Ω. We say that $\mathfrak{B}'$ *refines* $\mathfrak{B}$, denoted $\mathfrak{B}' \preceq \mathfrak{B}$, if for every $B' \in \mathfrak{B}'$ there is some $B \in \mathfrak{B}$ such that $B' \subseteq B$. If additionally $\mathfrak{B}' \neq \mathfrak{B}$ the partition $\mathfrak{B}'$ *strictly refines* $\mathfrak{B}$ which is denoted $\mathfrak{B}' \prec \mathfrak{B}$. Also, for a group $\Gamma \leq \mathrm{Sym}(\Omega)$ such that $\mathfrak{B}$ and $\mathfrak{B}'$ are Γ-invariant, let $\Gamma_B[\mathfrak{B}'[B]] \leq \mathrm{Sym}(\mathfrak{B}'[B])$ denote the induced natural action of Γ_B on $\mathfrak{B}'[B]$ for every $B \in \mathfrak{B}$. Finally, recall that a group $\Gamma \leq \mathrm{Sym}(\Omega)$ is semi-regular if the only element with fixed points is the identity.

▶ **Definition 7** (Almost d-ary Sequences of Partitions). *Let $\Gamma \leq \mathrm{Sym}(\Omega)$ be a group and let $\{\Omega\} = \mathfrak{B}_0 \succ \cdots \succ \mathfrak{B}_k = \{\{\alpha\} \mid \alpha \in \Omega\}$ be a sequence of Γ-invariant partitions. The sequence $\mathfrak{B}_0 \succ \cdots \succ \mathfrak{B}_k$ is* almost d-ary *if for every $i \in [k]$ and $B \in \mathfrak{B}_{i-1}$ it holds that*
1. *$|\mathfrak{B}_i[B]| \leq d$, or*
2. *$\Gamma_B[\mathfrak{B}_i[B]]$ is semi-regular.*

Since not every $\widehat{\Gamma}_d$-group $\Gamma \leq \mathrm{Sym}(\Omega)$ has an almost d-ary sequence of partitions, the first step of the algorithm is to normalize the input group. By a simple adaption of the arguments from [14] (see also [29]) we can obtain the following theorem normalizing the input to the Generalized String Isomorphism Problem.

▶ **Theorem 8.** *Let $(\Gamma, \mathfrak{P}, \mathfrak{X}, \mathfrak{Y})$ be an instance of the Generalized String Isomorphism Problem where $\Gamma \leq \mathrm{Sym}(\Omega)$ is a $\widehat{\Gamma}_d$-group.*

Then there is a set Ω^*, a monomorphism $\varphi\colon \Gamma \to \mathrm{Sym}(\Omega^*)$, an almost d-ary sequence of partitions $\mathfrak{B}_0^* \succ \mathfrak{B}_1^* \succ \cdots \succ \mathfrak{B}_\ell^*$ for Γ^φ, and an instance $(\Gamma^\varphi, \mathfrak{P}^*, \mathfrak{X}^*, \mathfrak{Y}^*)$ of the Generalized String Isomorphism Problem such that the following properties are satisfied:
1. *$|\Omega^*| \leq n^{\mathrm{f}_{norm}(d)+1}$ where $\mathrm{f}_{norm}(d) = \mathcal{O}(\log d)$,*
2. *there is some $i \in [k]$ such that $\mathfrak{P}^* = \mathfrak{B}_i^*$, and*
3. *$\gamma \in \mathrm{Iso}_\Gamma(\mathfrak{X}, \mathfrak{Y})$ if and only if $\varphi(\gamma) \in \mathrm{Iso}_{\Gamma^\varphi}(\mathfrak{X}^*, \mathfrak{Y}^*)$ for every $\gamma \in \Gamma$.*

Moreover, given $\Gamma \leq \mathrm{Sym}(\Omega)$, there is an algorithm computing the desired objects in time polynomial in the input size and the size of Ω^*.

Hence, in the following we restrict ourselves to the case where the input group has an almost d-ary sequence of partitions. In particular, this allows us to use the variant of the Unaffected Stabilizers Theorem proved in [14].

3.3 Creating Global Automorphisms from Local Information

The variant of the Unaffected Stabilizers Theorem from [14] provides us with the group-theoretic foundation to extend Babai's Local Certificates Routine [1] to the setting of hypergraphs. However, there is a second problem in generalizing the Local Certificates Routine that needs to be addressed.

Let $\Gamma \leq \mathrm{Sym}(\Omega)$ be a $\widehat{\Gamma}_d$-group, $\mathfrak{P}$ a Γ-invariant partition of Ω, and $\mathfrak{X}, \mathfrak{Y}$ two sets of $\mathfrak{P}$-strings. In a nutshell, the Local Certificates Routine considers a Γ-invariant window $W \subseteq \Omega$ such that $\Gamma[W] \leq \mathrm{Aut}(\mathfrak{X}[W])$ (i.e., the group Γ respects $\mathfrak{X}$ restricted to the window W) and aims at creating automorphisms of the entire structure $\mathfrak{X}$ (from the local information that Γ respects $\mathfrak{X}$ on the window W). In order to create these automorphisms the Local Certificates Routine considers the group $\Gamma_{(\Omega\setminus W)}$ fixing every point outside of W. Intuitively speaking, the Unaffected Stabilizers Theorem (resp. the variant suitable for $\widehat{\Gamma}_d$-groups) guarantees that the group $\Gamma_{(\Omega\setminus W)}$ is large. For the String Isomorphism Problem it is easy to see $\Gamma_{(\Omega\setminus W)}$ consists only of automorphisms of the input string since there are no dependencies between the positions within the window W and outside of W. However, for the Generalized String Isomorphism Problem, this is not true anymore. In other words, in order to compute $\mathrm{Aut}(\mathfrak{X})$, it is not possible to consider $\mathfrak{X}[W]$ and $\mathfrak{X}[\Omega \setminus W]$ independently.

Our solution to this problem is guided by the following simple observation. Suppose that $\mathfrak{X}[W]$ is simple, i.e., $m_{\mathfrak{X}[W]}(P) = 1$ for all $P \in \mathfrak{P}[W]$. In this case it is possible to consider $\mathfrak{X}[W]$ and $\mathfrak{X}[\Omega \setminus W]$ independently as the next lemma indicates.

▶ **Lemma 9.** *Let $\Gamma \leq \mathrm{Sym}(\Omega)$ be a permutation group, let $\mathfrak{P}$ be a Γ-invariant partition of Ω and $\mathfrak{X}$ a set of $\mathfrak{P}$-strings. Also suppose $W \subseteq \Omega$ is a Γ-invariant window such that $\mathfrak{X}[W]$ is simple and $\Gamma[W] \leq \mathrm{Aut}(\mathfrak{X}[W])$. Then $\Gamma_{(\Omega\setminus W)} \leq \mathrm{Aut}(\mathfrak{X})$.*

Proof. Let $\gamma \in \Gamma_{(\Omega\setminus W)}$ and $(P, \mathfrak{x}) \in \mathfrak{X}$. It suffices to show that $(P^\gamma, \mathfrak{x}^\gamma) \in \mathfrak{X}$. First suppose $P \subseteq W$. Then $(P, \mathfrak{x}) \in \mathfrak{X}[W]$ and $(P^\gamma, \mathfrak{x}^\gamma) \in \mathfrak{X}[W]$. Moreover, $P^\gamma \subseteq W$ since W is Γ-invariant. Hence, $(P^\gamma, \mathfrak{x}^\gamma) \in \mathfrak{X}$.

Otherwise $P \cap (\Omega\setminus W) \neq \emptyset$. Since $\alpha^\gamma = \alpha$ for all $\alpha \in \Omega\setminus W$ it follows that $P^\gamma \cap P \neq \emptyset$. Using the fact that $\mathfrak{P}$ is Γ-invariant this implies that $P^\gamma = P$. To complete the proof it is argued that $\mathfrak{x}^\gamma = \mathfrak{x}$. Let $\alpha \in P$. If $\alpha \notin W$ then $\mathfrak{x}^\gamma(\alpha) = \mathfrak{x}^\gamma(\alpha^\gamma) = \mathfrak{x}(\alpha^{\gamma\gamma^{-1}}) = \mathfrak{x}(\alpha)$. So assume $\alpha \in W$. Since $\gamma[W] \in \mathrm{Aut}(\mathfrak{X}[W])$ it holds that $((P \cap W)^\gamma, (\mathfrak{x}[P \cap W])^\gamma) = (P \cap W, (\mathfrak{x}[P \cap W])^\gamma) \in \mathfrak{X}[W]$. But this means $(\mathfrak{x}[P \cap W])^\gamma = \mathfrak{x}[P \cap W]$ since $\mathfrak{X}[W]$ is simple. Hence $\mathfrak{x}^\gamma(\alpha) = \mathfrak{x}(\alpha)$. ◀

Let $W \subseteq \Omega$ be a Γ-invariant set such that $\mathfrak{X}[W] \leq \mathrm{Aut}(\mathfrak{X}[W])$ (this is the case during the Local Certificates Routine). In order to solve the problem described above in general, the basic idea is to modify the instance in such a way that $\mathfrak{X}[W]$ becomes simple. This allows us to apply Lemma 9 and eventually, to extend the Local Certificates Routine to our setting. This modification is one of the key conceptual contributions of this work.

Consider a set $P \in \mathfrak{P}$. In order to "simplify" the instance we define an equivalence relation on the set $\mathfrak{X}[[P]]$ of all strings contained in P. Two $\mathfrak{P}$-strings $(P, \mathfrak{x}_1)$ and $(P, \mathfrak{x}_2)$ are W-equivalent if they are identical on the window W, i.e., $\mathfrak{x}_1[W] = \mathfrak{x}_2[W]$. For each equivalence class we create a new block P' containing exactly the strings from the equivalence class. Since the group Γ respects the induced subinstance $\mathfrak{X}[W]$ it naturally acts on the equivalence classes. This process is visualized in Figure 1 and formalized below.

Consider the natural homomorphism $\psi\colon \Gamma \to \mathrm{Sym}(\mathfrak{X}[W])$. Now let $\Omega' := \bigcup_{P \in \mathfrak{P}} P \times \{\mathfrak{x}[W \cap P] \mid (P, \mathfrak{x}) \in \mathfrak{X}\}$ and $\mathfrak{P}' := \{P \times \{\mathfrak{x}[W \cap P]\} \mid (P, \mathfrak{x}) \in \mathfrak{X}\}$. Also define

$$\mathfrak{X}' := \left\{ \Big(P \times \{\mathfrak{x}[W \cap P]\}, \mathfrak{x}'\colon P \times \{\mathfrak{x}[W \cap P]\} \to \Sigma\colon (\alpha, \mathfrak{x}[W \cap P]) \mapsto \mathfrak{x}(\alpha) \Big) \;\Big|\; (P, \mathfrak{x}) \in \mathfrak{X} \right\}$$

and similarly define $\mathfrak{Y}'$ for the instance $\mathfrak{Y}$. Note that $\mathfrak{X}'$ and $\mathfrak{Y}'$ are sets of $\mathfrak{P}'$-strings. Finally, the group Γ faithfully acts on the set Ω' via $(\alpha, \mathfrak{z})^\gamma = (\alpha^\gamma, \mathfrak{z}^\gamma)$ yielding an injective homomorphism $\psi\colon \Gamma \to \mathrm{Sym}(\Omega')$. Define $\Gamma' := \Gamma^\psi$. It can be easily checked that the updated instance is equivalent to the original instance. Also, $\mathfrak{X}'[W']$ is simple where $W' := \{(\alpha, \mathfrak{z}) \in \Omega' \mid \alpha \in W\}$.

While this simplification allows us to treat $\mathfrak{X}'[W']$ and $\mathfrak{X}'[W' \setminus \Omega']$ independently and thus solves the above problem, it creates several additional issues that need to be addressed. First, this modification may destroy the normalization property (i.e., the existence of an

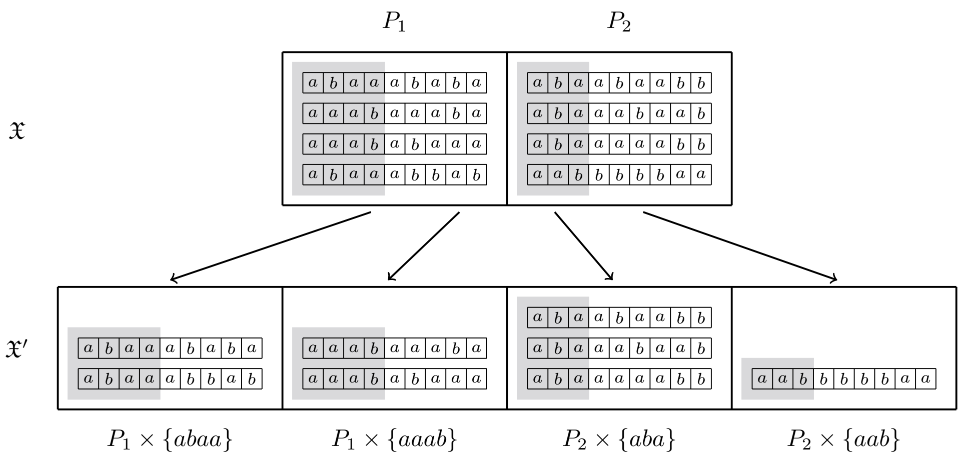

Figure 1 A set $\mathfrak{X}$ of $\mathfrak{P}$-strings is given in the top and the "simplified" instance $\mathfrak{X}'$ is given below. The window W is marked in gray. Note that $\mathfrak{X}'[W']$ is simple where W' denotes the window marked in gray in the bottom part of the figure.

almost d-ary sequence of partitions). As a result, the Local Certificates Routine constantly needs to renormalize the input instances which requires a precise analysis of the increase in size occurring from the renormalization (see Theorem 8). In order to be able to still analyze the progress made by the recursion, we introduce the notion of a *virtual size* of an instance.

▶ **Definition 10.** *Let $\mathfrak{P}$ be a partition of Ω and suppose $\mathfrak{X}$ is a set of $\mathfrak{P}$-strings. The d-*virtual size *of $\mathfrak{X}$ is defined by $s := \sum_{P\in\mathfrak{P}} |P| \cdot (m_{\mathfrak{X}}(P))^{\mathsf{f}_{\mathsf{norm}}(d)+1}$.*

Here, $\mathsf{f}_{\mathsf{norm}}(d) = \mathcal{O}(\log d)$ refers to the normalization cost defined in Theorem 8. Hence, the d-virtual size s of $\mathfrak{X}$ is bounded by $(m+n)^{\mathcal{O}(\log d)}$ and it suffices to analyze the running time of all subroutines in terms of the virtual size of the input instances. Intuitively speaking, the idea of the virtual size is to take into account the cost of all potential renormalization steps which means that, when the algorithm renormalizes an instance, while its actual size might grow significantly, its virtual size does not. This allows us to measure the progress made by the recursive algorithm although instances might grow significantly.

The renormalization of instances also creates another problem. In the *aggregation of local certificates*, the outputs of the Local Certificates Routine are compared with each other requiring the outputs to be isomorphism-invariant. However, the renormalization procedure is not isomorphism-invariant. Our solution to this problem is to run the Local Certificates Routine in parallel on all pairs of test sets compared later on. This way, we can ensure that all instances are normalized in the same way.

This simplification procedure is formalized by the next technical theorem which combines all the requirements outlined above, including the renormalization steps, into a single statement.

▶ **Theorem 11.** *Let $\Gamma \leq \mathrm{Sym}(\Omega)$ be a $\widehat{\Gamma}_d$-group and $\mathfrak{P}$ be a Γ-invariant partition of Ω. Also suppose $\{\Omega\} = \mathfrak{B}_0 \succ \mathfrak{B}_1 \succ \cdots \succ \mathfrak{B}_\ell = \{\{\alpha\} \mid \alpha \in \Omega\}$ forms an almost d-ary sequence of Γ-invariant partitions such that $\mathfrak{P} = \mathfrak{B}_i$ for some $i \in [\ell]$. Let $(\mathfrak{X}_i)_{i\in[p]}$ be a list of sets of $\mathfrak{P}$-strings. Also let $W \subseteq \Omega$ be a Γ-invariant set such that $\Gamma[W] \leq \mathrm{Aut}(\mathfrak{X}_i[W])$ for all $i \in [p]$. Moreover, assume that $\mathfrak{X}_i[W] = \mathfrak{X}_j[W]$ for all $i, j \in [p]$.*

Then there is an equivalence relation $\sim$ on the set $[p]$ such that $\mathfrak{X}_i \cong_\Gamma \mathfrak{X}_j$ implies $i \sim j$ for all $i, j \in [p]$, and for each equivalence class $A \subseteq [p]$, we get the following:

A set Ω^*, a group $\Gamma^* \leq \mathrm{Sym}(\Omega)$, elements $\lambda_i \in \Gamma$ for all $i \in A$, a monomorphism $\varphi\colon \Gamma^* \to \Gamma$, a window $W^* \subseteq \Omega^*$, a sequence of partitions $\{\Omega^*\} = \mathfrak{B}_0^* \succ \mathfrak{B}_1^* \succ \cdots \succ \mathfrak{B}_k^* = \{\{\alpha\} \mid \alpha \in \Omega^*\}$, and a list of $\mathfrak{P}^*$-strings $(\mathfrak{X}_i^*)_{i\in A}$, such that the following properties are satisfied:

(A) *the sequence $\mathfrak{B}_0^* \succ \cdots \succ \mathfrak{B}_k^*$ forms an almost d-ary sequence of Γ^*-invariant partitions,*
(B) *W^* is Γ^*-invariant and $\Gamma^*[W^*] \leq \mathrm{Aut}(\mathfrak{X}_i^*[W^*])$ for all $i \in A$,*
(C) *there is some $i \in [k]$ such that $\mathfrak{P}^* = \mathfrak{B}_i^*$,*
(D) *$\mathrm{Iso}_\Gamma(\mathfrak{X}_i, \mathfrak{X}_j) = \lambda_i^{-1}(\mathrm{Iso}_{\Gamma^*}(\mathfrak{X}_i^*, \mathfrak{X}_j^*))^\varphi \lambda_j$ for all $i, j \in A$,*
(E) *$\mathfrak{X}_i^*[W^*]$ is simple for all $i \in A$,*
(F) *for s the virtual size of $\mathfrak{X}_i$ and s^* the virtual size of $\mathfrak{X}_i^*$ it holds that $s^* \leq s$, and*
(G) *for s the virtual size of $\mathfrak{X}_i[\Omega \setminus W]$ and s^* the virtual size of $\mathfrak{X}_i^*[\Omega^* \setminus W^*]$ it holds that $s^* \leq s$.*

Moreover, there is an algorithm computing the desired objects in time $p^2 \cdot (n+m)^{\mathcal{O}((\log d)^c)}$ for some absolute constant c.

3.4 The Local Certificates Routine

In this subsection the Local Certificates Routine originally introduced in [1] is lifted to the Generalized String Isomorphism Problem for $\widehat{\Gamma}_d$-group which builds the crucial step of extending the group-theoretic techniques of Babai's quasipolynomial time isomorphism test to the setting of this paper.

Let $\Gamma \leq \mathrm{Sym}(\Omega)$ be a permutation group and let $(\Gamma, \mathfrak{P}, \mathfrak{X}, \mathfrak{Y})$ be an instance of the Generalized String Isomorphism Problem. Furthermore let $\varphi\colon \Gamma \to S_k$ be a *giant representation*, i.e., a homomorphism $\varphi\colon \Gamma \to S_k$ such that $\Gamma^\varphi \geq A_k$. For the description of the Local Certificates Routine we extend the notation of set- and point-wise stabilizers for the group Γ to the action on the set $[k]$ defined via the giant representation φ. For a set $T \subseteq [k]$ let $\Gamma_T := \varphi^{-1}((\Gamma^\varphi)_T)$ and $\Gamma_{(T)} := \varphi^{-1}((\Gamma^\varphi)_{(T)})$.

The basic approach of the Local Certificates Routine is to consider *test sets* $T \subseteq [k]$ of logarithmic size.

▶ **Definition 12.** *A test set $T \subseteq [k]$ is* full *if $(\mathrm{Aut}_{\Gamma_T}(\mathfrak{X}))^\varphi[T] \geq \mathrm{Alt}(T)$. A* certificate of fullness *is a subgroup $\Delta \leq \mathrm{Aut}_{\Gamma_T}(\mathfrak{X})$ such that $\Delta^\varphi[T] \geq \mathrm{Alt}(T)$. A* certificate of non-fullness *is a non-giant $\Lambda \leq \mathrm{Sym}(T)$ such that $(\mathrm{Aut}_{\Gamma_T}(\mathfrak{X}))^\varphi[T] \leq \Lambda$.*

The central part of the algorithm is to determine for each test set $T \subseteq [k]$ (of size t approximately logarithmic in d) whether T is full and, depending on the outcome, compute a certificate of fullness or a certificate of non-fullness. Actually, in order to decide isomorphism, non-fullness certificates are also required for pairs of test sets. All of this is achieved by the following theorem.

▶ **Theorem 13.** *Let $\mathfrak{P}$ be a partition of Ω, $\mathfrak{X}$ and $\mathfrak{Y}$ two sets of $\mathfrak{P}$-strings. Let s be the d-virtual size of $\mathfrak{X}$ and $\mathfrak{Y}$. Also let $\Gamma \leq \mathrm{Sym}(\Omega)$ a $\widehat{\Gamma}_d$-group that has an almost d-ary sequence of partitions $\mathfrak{B}_0 \succ \cdots \succ \mathfrak{B}_\ell$ such that $\mathfrak{P} = \mathfrak{B}_i$ for some $i \in [\ell]$. Furthermore suppose there is a giant representation $\varphi\colon \Gamma \to S_k$ and let $k \geq t > \max\{8, 2 + \log_2 d\}$. Finally, define $\mathcal{T} = \{(\mathfrak{X}, T), (\mathfrak{Y}, T) \mid T \subseteq [k], |T| = t\}$. For every $(\mathfrak{Z}_1, T_1), (\mathfrak{Z}_2, T_2) \in \mathcal{T}$ one can compute*

(i) *if T_1 is full with respect to $\mathfrak{Z}_1$, a group $\Delta := \Delta(\mathfrak{Z}_1, T_1) \leq \mathrm{Aut}_{\Gamma_{T_1}}(\mathfrak{Z}_1)$ such that $\Delta^\varphi[T_1] \geq \mathrm{Alt}(T_1)$, or*
(ii) *if T_1 is not full with respect to $\mathfrak{Z}_1$, a non-giant group $\Lambda := \Lambda(T_1, \mathfrak{Z}_1, T_2, \mathfrak{Z}_2) \leq \mathrm{Sym}(T_1)$ and a bijection $\lambda := \lambda(T_1, \mathfrak{Z}_1, T_2, \mathfrak{Z}_2)\colon T_1 \to T_2$ such that*

$$\left\{\gamma^\varphi|_{T_1} \;\middle|\; \gamma \in \mathrm{Iso}_\Gamma(\mathfrak{Z}_1, \mathfrak{Z}_2) \wedge T_1^{(\gamma^\varphi)} = T_2\right\} \subseteq \Lambda\lambda.$$

Moreover, the output is isomorphism-invariant, i.e., for every two pairs $(\mathfrak{Z}_1, T_1), (\mathfrak{Z}_2, T_2) \in \mathcal{T}$ *and* $(\mathfrak{Z}'_1, T'_1), (\mathfrak{Z}'_2, T'_2) \in \mathcal{T}$ *and isomorphisms* $\gamma_i \in \mathrm{Iso}_\Gamma((\mathfrak{Z}_i, T_i), (\mathfrak{Z}'_i, T'_i))$, $i \in \{1, 2\}$, *it holds that*

(a) *if* T_1 *is full with respect to* $\mathfrak{Z}_1$, *then* $(\Delta(T_1, \mathfrak{Z}_1))^{\gamma_1} = \Delta(T'_1, \mathfrak{Z}'_1)$, *and*

(b) *if* T_1 *is not full with respect to* $\mathfrak{Z}_1$, *then*

$$\varphi(\gamma_1^{-1})\Lambda(T_1, \mathfrak{Z}_1, T_2, \mathfrak{Z}_2)\lambda(T_1, \mathfrak{Z}_1, T_2, \mathfrak{Z}_2)\varphi(\gamma_2) = \Lambda(T'_1, \mathfrak{Z}'_1, T'_2, \mathfrak{Z}'_2)\lambda(T'_1, \mathfrak{Z}'_1, T'_2, \mathfrak{Z}'_2).$$

Moreover, there are numbers $s_1, \dots, s_r \leq s/2$ *such that* $\sum_{i=1}^{r} s_i \leq 4k^{2t} \cdot t! \cdot s$ *and, for each* $i \in [r]$ *using a recursive call to the Generalized String Isomorphism Problem for instances of* d*-virtual size at most* s_i, *and* $k^{\mathcal{O}(t)} \cdot t! \cdot (n+m)^{\mathcal{O}((\log d)^c)}$ *additional computation steps, an algorithm can compute all desired objects.*

The algorithm is similar to the standard Local Certificates Routine implement in Babai's quasipolynomial-time isomorphism test [1]. A main difference is that, in each iteration, the algorithm runs the subroutine implemented in Theorem 11 to “simplify” all instances as described in the previous subsection. This guarantees that, throughout the execution of the algorithm, the prerequisites of Lemma 9 are satisfied allowing for the construction of global automorphisms from local information.

3.5 An Algorithm for the Generalized String Isomorphism Problem

With the Local Certificates Routine for sets of $\mathfrak{P}$-strings presented above it is possible to provide an algorithm for the Hypergraph Isomorphism Problem assuming the input group is equipped with an almost d-ary sequence of partitions. The algorithm mostly follows the same patterns as in [14] giving the corresponding result for the String Isomorphism Problem by replacing the Local Certificates Routine.

▶ **Theorem 14.** *There is an algorithm that, given a partition* $\mathfrak{P}$ *of* Ω, *a* $\widehat{\Gamma}_d$*-group* $\Gamma \leq \mathrm{Sym}(\Omega)$, *two sets of* $\mathfrak{P}$*-strings* $\mathfrak{X}, \mathfrak{Y}$ *and an almost* d*-ary sequence of partitions* $\mathfrak{B}_0 \succ \dots \succ \mathfrak{B}_\ell$ *for* Γ *such that* $\mathfrak{P} = \mathfrak{B}_i$ *for some* $i \in [\ell]$, *computes a representation for* $\mathrm{Iso}_\Gamma(\mathfrak{X}, \mathfrak{Y})$ *in time* $(n+m)^{\mathcal{O}((\log d)^c)}$, *for an absolute constant* c.

Combining Theorem 8 and 14 gives the main technical result of this work.

▶ **Theorem 15.** *The Generalized String Isomorphism Problem for* $\widehat{\Gamma}_d$*-groups can be solved in time* $(n+m)^{\mathcal{O}((\log d)^c)}$ *for some constant* c.

As an immediate consequence we obtain one of the main results of this paper.

▶ **Corollary 16** (Theorem 1 restated)**.** *The Hypergraph Isomorphism Problem for* $\widehat{\Gamma}_d$*-groups can be solved in time* $(n+m)^{\mathcal{O}((\log d)^c)}$ *for some constant* c.

4 Color Refinement and Small Color Classes

In the following two sections we present applications of the improvement obtained for the Hypergraph Isomorphism Problem for $\widehat{\Gamma}_d$-groups. Towards this end, we first introduce the notion of t-CR-bounded graphs and present an isomorphism test for such graphs running in time $n^{\mathcal{O}((\log t)^c)}$. The notion of t-CR-bounded graphs generalizes color-t-bounded graphs considered in [4] in the context of the isomorphism problem for strongly regular graphs. But more importantly, the isomorphism problem for t-CR-bounded graphs can serve as an intermediate problem for designing faster isomorphism tests for important classes of graphs.

For example, there is a simple polynomial-time Turing reduction from the isomorphism problem for graphs of maximum degree d to the isomorphism problem for d-CR-bounded graphs. Hence, our results may be seen as a generalization to the isomorphism test for graphs of small degree presented in [14]. Moreover, in the next section, we present a polynomial-time Turing reduction from the isomorphism problem for graphs of genus at most g to the isomorphism problem for $(4g+2)$-CR-bounded graphs. As a result, isomorphism for graphs of genus at most g can be tested in time $n^{\mathcal{O}((\log g)^c)}$.

The definition of t-CR-bounded graphs builds on the Color Refinement algorithm, a simple combinatorial algorithm that iteratively refines a vertex-coloring in an isomorphism-invariant manner and which forms a fundamental algorithmic tool in the context of the Graph Isomorphism Problem (see, e.g., [4, 7, 8, 24, 25, 36]). We start by formally defining the outcome of the Color Refinement algorithm in the next subsection.

4.1 The Color Refinement Algorithm

Let G be a graph with vertex coloring $\chi_V\colon V(G) \to C_V$ and arc coloring $\chi_E\colon \{(v,w) \mid vw \in E(G)\} \to C_E$. The *Color Refinement algorithm* is a procedure that, given a vertex- and arc-colored graph G, iteratively computes an isomorphism-invariant refinement $\chi_{\mathsf{CR}}[G]$ of the vertex-coloring χ_V.

Let $\chi_1, \chi_2\colon V \to C$ be colorings of vertices where C is some finite set of colors. The coloring χ_1 *refines* χ_2, denoted $\chi_1 \preceq \chi_2$, if $\chi_1(v) = \chi_1(w)$ implies $\chi_2(v) = \chi_2(w)$ for all $v, w \in V$. The colorings χ_1 and χ_2 are *equivalent*, denoted $\chi_1 \equiv \chi_2$, if $\chi_1 \preceq \chi_2$ and $\chi_2 \preceq \chi_1$.

Given a vertex- and arc-colored graph G, the Color Refinement algorithm computes an isomorphism-invariant coloring $\chi_{\mathsf{CR}}[G]$ as follows. The initial coloring for the algorithm is defined as $\chi_{(0)}[G] \coloneqq \chi_V$, the vertex-coloring of the input graph. The initial coloring is refined by iteratively computing colorings $\chi_{(i)}[G]$ for $i > 0$. For $i > 0$ define $\chi_{(i)}[G](v) \coloneqq (\chi_{(i-1)}[G](v), \mathcal{M}_i(v))$ where

$$\mathcal{M}_i(v) \coloneqq \{\{\big(\chi_{(i-1)}[G](w), \chi_E(v,w), \chi_E(w,v)\big) \mid w \in N_G(v)\}\}.$$

From the definition of the colorings it is immediately clear that $\chi_{(i+1)}[G] \preceq \chi_{(i)}[G]$. Now let $i \in \mathbb{N}$ be the minimal number such that $\chi_{(i)}[G] \equiv \chi_{(i+1)}[G]$. For this i, the coloring $\chi_{(i)}[G]$ is called the *stable* coloring of G and is denoted by $\chi_{\mathsf{CR}}[G]$.

The *Color Refinement algorithm* takes as input a (vertex- and arc-colored) graph G and computes (a coloring that is equivalent to) $\chi_{\mathsf{CR}}[G]$. I remark that this can be implemented in time almost linear in the number of vertices and edges (see, e.g., [9]).

4.2 Splitting Small Color Classes

Having defined the Color Refinement algorithm, we can now define the notion of t-CR-bounded graphs. The basic idea behind t-CR-bounded graphs is the following. Suppose (G, χ) is a vertex-colored graph. Then (G, χ) is t-CR-bounded if it is possible to transform χ into a discrete coloring (i.e., a coloring where each vertex has its own color) by repeatedly performing the following two operations: applying the Color Refinement algorithm and completely splitting color classes of size at most t (i.e., assigning every vertex in such a color class its own color).

For formal reasons, we actually define t-CR-bounded pairs where an additional set $S \subseteq V(G)$ is provided each vertex of which may also be individualized. The intuition behind this is that we may already have good knowledge about the structure of the automorphism group of (G, χ) on the set S which can be exploited for isomorphism testing.

▶ **Definition 17.** *A pair (G, S), where $G = (V, E, \chi_V, \chi_E)$ is a vertex- and arc-colored graph and $S = \chi_V^{-1}(c)$ for some color c (not necessarily in the image of χ_V, i.e., S may be the empty set), is* t-CR-bounded *if the sequence $(\chi_i)_{i\geq 0}$ reaches a discrete coloring where*

$$\chi_0(v) := \begin{cases} (v, 1) & \text{if } v \in S \\ (\chi_V(v), 0) & \text{if } v \notin S \end{cases},$$

$\chi_{2i+1} := \chi_{CR}[V, E, \chi_{2i}, \chi_E]$ *and*

$$\chi_{2i+2}(v) := \begin{cases} (v, 1) & \text{if } |\chi_{2i+1}^{-1}(\chi_{2i+1}(v))| \leq t \\ (\chi_{2i+1}(v), 0) & \text{otherwise} \end{cases}$$

for all $i \geq 0$. A graph G is t-CR-bounded *if the pair $(G, \emptyset)$ is t-CR-bounded.*

I remark that the name "t-CR-bounded" is inspired by color-t-bounded graphs [4] defined in a similar manner where the letters *CR* refer to the Color Refinement algorithm. Also, I note that a similar notion of graphs has been exploited in [28] for designing a polynomial-time isomorphism test for unit square graphs.

▶ **Theorem 18.** *Let (G_1, S_1) and (G_2, S_2) be two t-CR-bounded pairs and also let $\Gamma \leq \mathrm{Sym}(S_1)$ be a $\widehat{\Gamma}_t$-group and $\theta\colon S_1 \to S_2$ a bijection. Then a representation for the set*

$$\mathrm{Iso}_{\Gamma\theta}((G_1, S_1), (G_2, S_2)) := \{\sigma\colon G_1 \cong G_2 \mid \sigma|_{S_1} \in \Gamma\theta\}$$

can be computed in time $n^{\mathcal{O}((\log t)^c)}$ for some absolute constant c.

▶ **Corollary 19.** *The Graph Isomorphism Problem for t-CR-bounded graphs can be solved in time $n^{\mathcal{O}((\log t)^c)}$ for some absolute constant c.*

In particular, this includes color-t-bounded graphs considered in [4] in the context of the isomorphism problem for strongly regular graphs.

The algorithm underlying Theorem 18 actually describes a polynomial-time Turing reduction from the isomorphism problem for t-CR-bounded graphs to the Hypergraph Isomorphism Problem for $\widehat{\Gamma}_t$-groups. This result may be of independent interest.

▶ **Lemma 20.** *There is a polynomial-time Turing reduction from the Graph Isomorphism Problem for t-CR-bounded graphs to the Hypergraph Isomorphism Problem for $\widehat{\Gamma}_t$-groups.*

4.3 Hereditarily Finite Sets

As the first application to the isomorphism problem for t-CR-bounded pairs we consider hereditarily finite sets. Intuitively speaking, a hereditarily finite set over ground set V is any type of combinatorial object that can be build by iteratively taking sets and tuples over previously created objects.

In this section we extend the algorithm for solving the Hypergraph Isomorphism Problem for $\widehat{\Gamma}_d$-groups to work on any type of hereditarily finite set.

Let V be finite set of elements (e.g., the vertex of a graph). The class of *hereditarily finite sets* over universe (ground set) V is inductively defined as follows. Each $v \in V$ is an *atom* which in particular is a hereditarily finite set. Also, for hereditarily finite sets $\mathfrak{A}_1, \dots, \mathfrak{A}_k$ (over universe V) the set $\{\mathfrak{A}_1, \dots, \mathfrak{A}_k\}$ as well as the tuple $(\mathfrak{A}_1, \dots, \mathfrak{A}_k)$ is a hereditarily finite set (over universe V). Examples of hereditarily finite sets include graphs, hypergraphs and relational structures.

In order to apply the isomorphism test for t-CR-bounded graphs, we can transform a hereditarily finite set $\mathfrak{A}$ over ground set V into a graph $G(\mathfrak{A})$ in the natural way so that the pair $(G(\mathfrak{A}), V)$ is 0-CR-bounded.

▶ **Corollary 21.** *Let $\mathfrak{A}_1$ and $\mathfrak{A}_2$ be two hereditarily finite sets over the universe V and let $\Gamma \leq \mathrm{Sym}(V)$ be a $\widehat{\Gamma}_d$-group. Then a representation for the set $\mathrm{Iso}_\Gamma(\mathfrak{A}_1, \mathfrak{A}_2) := \{\gamma \in \Gamma \mid \gamma\colon \mathfrak{A}_1 \cong \mathfrak{A}_2\}$ can be computed in time $(n+m)^{\mathcal{O}((\log d)^c)}$ for some absolute constant c where $n := |V|$ and $m := |V(G(\mathfrak{A}_1))|$.*

▶ **Corollary 22.** *The isomorphism problem for hereditarily finite sets can be solved in time $(n+m)^{\mathcal{O}((\log n)^c)}$.*

I remark that a variant of the previous corollary was independently obtained by Daniel Wiebking [37] (see *Related Work*).

5 Isomorphism for Graphs of Bounded Genus

Next, we present a second, slightly more involved application of our results and give an algorithm solving the isomorphism problem for graphs of Euler genus at most g in time $n^{\mathcal{O}((\log g)^c)}$ for some absolute constant c. Actually, we prove a more general result.

Recall that a graph H is a minor of another graph G if H can be obtained from G by removing vertices as well as removing and contracting edges. Also define $K_{m,n}$ to be the complete bipartite graph with m vertices on the left side and n vertices on the right side. Let $h \geq 3$ and define $\mathcal{C}_h$ to be the class of graphs that exclude $K_{3,h}$ as a minor.

We present a polynomial-time reduction from the isomorphism problem for the class $\mathcal{C}_h$ to the isomorphism problem for t-CR-bounded graphs where $t := h - 1$. The following lemma is the key tool for the reduction. Intuitively, it investigates the structure of certain colorings that are stable with respect to the Color Refinement algorithm for graphs in the class $\mathcal{C}_h$. Recall that a graph G is 3-*connected* if there are no two vertices $v, w \in V(G)$ such that $G - \{v, w\}$ is disconnected.

▶ **Lemma 23.** *Let (G, χ) be a 3-connected, colored graph that excludes $K_{3,h}$ as a minor and suppose $V_1 \uplus V_2 = V(G)$ such that*

1. *each $v \in V_1$ forms a singleton color class with respect to χ,*
2. *χ is stable with respect to the Color Refinement algorithm,*
3. *$|V_1| \geq 3$, and*
4. *$N(V_2) = V_1$.*

Then there is a color class $U \subseteq V_2$ with respect to χ of size $|U| \leq h - 1$.

Proof. Let $C := \mathrm{im}(\chi)$, $C_1 := \chi(V_1)$ and $C_2 := \chi(V_2)$. Also define H to be the graph with vertex set $V(H) := C$ and edge set

$$E(H) = \{c_1c_2 \mid \exists v_1 \in \chi^{-1}(c_1), v_2 \in \chi^{-1}(c_2)\colon v_1v_2 \in E(G)\}.$$

Let $C' \subseteq C_2$ be the vertex set of a connected component of $H[C_2]$. Then $|N_H(C')| \geq 3$ since each $v \in V_1$ forms a singleton color class with respect to χ and G is 3-connected.

Now let $c_1, c_2, c_3 \in N_H(C')$ be distinct and also let $v_i \in \chi^{-1}(c_i)$ for $i \in [3]$. Also let T be a spanning tree of $H[C' \cup \{c_1, c_2, c_3\}]$ such that $c_1, c_2, c_3 \in L(T)$ where $L(T)$ denotes the set of leaves of T. Moreover, let T' be the subtree of T obtained from repeatedly removing all leaves $c \in C'$. Hence, $L(T') = \{c_1, c_2, c_3\}$. Then there is a unique color c such that $\deg_{T'}(c) = 3$. Also, for $i \in [3]$, define C_i' to be the set of internal vertices on the unique path from c_i to c in the tree T'.

Since $|\chi^{-1}(c_i)| = 1$ and χ is stable with respect to the Color Refinement algorithm it holds that

$$G\left[\chi^{-1}(C'_i \cup \{c_i\})\right]$$

is connected. Let $U_i := \chi^{-1}(C'_i \cup \{c_i\})$, $i \in [3]$. Also let $U = \chi^{-1}(c)$ and suppose that $|U| \geq h$. Then $N(U_i) \cap U \neq \emptyset$ by the definition of the tree T. Moreover, this implies $U \subseteq N(U_i)$ since χ is stable with respect to the Color Refinement algorithm. Hence, G contains a minor isomorphic to $K_{3,h}$. ◀

▶ **Corollary 24.** *Let $(G, \chi_V, \chi_E) \in \mathcal{C}_h$ be a 3-connected, vertex- and arc-colored graph and let $v_1, v_2, v_3 \in V(G)$. Also define $\chi_V^*(v_i) := (i, 1)$ for $i \in [3]$ and $\chi_V^*(v) := (\chi_V(v), 0)$ for all $v \in V(G) \setminus \{v_1, v_2, v_3\}$. Then (G, χ_V^*, χ_E) is $(h-1)$-CR-bounded.*

Proof. Let $(\chi_i)_{i\geq 0}$ be the sequence of colorings obtained from the definition of $(h-1)$-CR-bounded graphs (Definition 17) for the graph (G, χ_V^*, χ_E). Let $\chi^* := \chi_i$ for the minimal $i \geq 0$ such that $\chi_i \equiv \chi_{i+1}$.

Suppose towards a contradiction that χ^* is not discrete (i.e., not every color class is a singleton). Let $V_2 := \{v \in V(G) \mid |(\chi^*)^{-1}(\chi^*(v))| > 1\}$ and let $V_1 := N_G(V_2)$. Then $|V_1| \geq 3$ since $|V(G) \setminus V_2| \geq 3$ and G is 3-connected. Also note that $\chi^*|_{V_1 \cup V_2}$ is a stable coloring for the graph $G[V_1 \cup V_2]$. Hence, by Lemma 23, there is some color c such that $1 < |(\chi^*)^{-1}(c)| \leq h-1$. But this contradicts the definition of the coloring χ^* (cf. Definition 17). ◀

The last corollary gives some insights into the structure of the automorphism group of graphs $G \in \mathcal{C}_h$.

▶ **Theorem 25.** *Let $G \in \mathcal{C}_h$ be a 3-connected graph and let $v_1, v_2, v_3 \in V(G)$ be distinct vertices. Then $(\mathrm{Aut}(G))_{(v_1,v_2,v_3)}$ is a $\widehat{\Gamma}_{h-1}$-group.*

Also, the corollary can be used to design an isomorphism test for the class $\mathcal{C}_h$.

▶ **Corollary 26** (Theorem 3 restated)**.** *The Graph Isomorphism Problem for the class $\mathcal{C}_h$ can be solved in time $n^{\mathcal{O}((\log h)^c)}$ for some absolute constant c.*

Proof. Let $G_1, G_2 \in \mathcal{C}_h$. By standard decomposition techniques it suffices to consider the case where G_1 and G_2 are (vertex- and arc-colored) 3-connected graphs.

Suppose $G_1 = (V_1, E_1, \chi_V^1, \chi_E^1)$ and $G_2 = (V_2, E_2, \chi_V^2, \chi_E^2)$. Let $v_1, v_2, v_3 \in V(G_1)$ be three arbitrary vertices. For every $w_1, w_2, w_3 \in V(G_2)$ it is checked whether there is some isomorphism $\varphi\colon G_1 \cong G_2$ such that $\varphi(v_i) = w_i$ for all $i \in [3]$. Towards this end, define $\widehat{\chi}_V^1(v_i) = (i, 1)$ for $i \in [3]$ and $\widehat{\chi}_V^1(v) = (\chi_V^1(v), 0)$ for all $v \in V(G_1) \setminus \{v_1, v_2, v_3\}$. Similarly, define $\widehat{\chi}_V^2(w_i) = (i, 1)$ for $i \in [3]$ and $\widehat{\chi}_V^2(w) = (\chi_V^2(w), 0)$ for all $w \in V(G_2) \setminus \{w_1, w_2, w_3\}$. Hence, it needs to be checked whether $\widehat{G}_1 \cong \widehat{G}_2$ where $\widehat{G}_j := (V_j, E_j, \widehat{\chi}_V^j, \chi_E^j)$, $j \in [2]$. By Corollary 24 the graphs $\widehat{G}_1$ and $\widehat{G}_2$ are $(h-1)$-CR-bounded. Hence, isomorphism of the two graphs can be tested within the desired time by Corollary 19. ◀

Note that the algorithm from the corollary describes a polynomial-time Turing reduction from the Graph Isomorphism Problem for $\mathcal{C}_h$ to the Graph Isomorphism Problem for $(h-1)$-CR-bounded graphs. In combination with Lemma 20 this means the the Graph Isomorphism Problem for $\mathcal{C}_h$ is polynomial-time Turing reducible to the Hypergraph Isomorphism Problem for $\widehat{\Gamma}_{h-1}$-groups.

Since the class of graphs of Euler genus at most g excludes $K_{3,4g+3}$ as a minor [32, 18], we obtain the following result.

▶ **Corollary 27** (Corollary 4 restated)**.** *The Graph Isomorphism Problem for graphs of genus at most g can be solved in time $n^{\mathcal{O}((\log g)^c)}$ for some absolute constant c.*

6 Conclusion

We provided a faster algorithm for the Hypergraph Isomorphism Problem for $\widehat{\Gamma}_d$-groups running in time $(n+m)^{\mathcal{O}((\log d)^c)}$ for some absolute constant c. As an application, we obtained, for example, an algorithm testing isomorphism of graphs excluding $K_{3,h}$ as a minor in time $n^{\mathcal{O}((\log h)^c)}$. In particular, this gives an isomorphism test for graphs of Euler genus at most g running in time $n^{\mathcal{O}((\log g)^c)}$.

With the Hypergraph Isomorphism Problem for $\widehat{\Gamma}_d$-groups being exploited as a subroutine in a number of algorithms testing isomorphism, it seems plausible to hope for further applications beyond the ones presented in this paper. Indeed, a very recent work by Grohe, Wiebking and the present author [16] gives an isomorphism test running in time $n^{\mathcal{O}((\log h)^c)}$ for n-vertex graphs excluding an arbitrary h-vertex graph as a minor. This algorithm crucially builds on the isomorphism test for hypergraphs as well as the notion of t-CR-bounded graphs. Actually, extending the applicability of our techniques further, it might be possible to provide an algorithm with a similar running time for all classes excluding only a topological minor building on a decomposition theorem for such graph classes due to Grohe and Marx [13].

Another open question concerns the complexity of testing isomorphism of hypergraphs for a given $\widehat{\Gamma}_d$-group. The algorithm presented in this work is significantly faster than the previous best algorithm [34] running in time $n^{\mathcal{O}(d)}m^{\mathcal{O}(1)}$ only for small numbers of hyperedges. Indeed, for large numbers of hyperedges $m = n^{\Omega(d)}$, our algorithm becomes slower than the algorithm due to Schweitzer and Wiebking [34]. Can the Hypergraph Isomorphism Problem for $\widehat{\Gamma}_d$-groups be solved in time $n^{\mathcal{O}((\log d)^c)}m^{\mathcal{O}(1)}$ for some absolute constant c? Observe that it is already completely unclear whether isomorphism of hypergraphs can be tested in time $n^{\mathcal{O}((\log n)^c)}m^{\mathcal{O}(1)}$ for some absolute constant c.

References

1 László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 684–697. ACM, 2016. doi:10.1145/2897518.2897542.

2 László Babai. Canonical form for graphs in quasipolynomial time: preliminary report. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1237–1246. ACM, 2019. doi:10.1145/3313276.3316356.

3 László Babai, Peter J. Cameron, and Péter P. Pálfy. On the orders of primitive groups with restricted nonabelian composition factors. *J. Algebra*, 79(1):161–168, 1982. doi:10.1016/0021-8693(82)90323-4.

4 László Babai, Xi Chen, Xiaorui Sun, Shang-Hua Teng, and John Wilmes. Faster canonical forms for strongly regular graphs. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 157–166. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.25.

5 László Babai and Paolo Codenotti. Isomorhism of hypergraphs of low rank in moderately exponential time. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 667–676. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.80.

6 László Babai, William M. Kantor, and Eugene M. Luks. Computational complexity and the classification of finite simple groups. In *24th Annual Symposium on Foundations of Computer Science, Tucson, Arizona, USA, 7-9 November 1983*, pages 162–171. IEEE Computer Society, 1983. doi:10.1109/SFCS.1983.10.

7 László Babai and Eugene M. Luks. Canonical labeling of graphs. In David S. Johnson, Ronald Fagin, Michael L. Fredman, David Harel, Richard M. Karp, Nancy A. Lynch, Christos H. Papadimitriou, Ronald L. Rivest, Walter L. Ruzzo, and Joel I. Seiferas, editors, *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 171–183. ACM, 1983. doi:10.1145/800061.808746.

8 László Babai and John Wilmes. Quasipolynomial-time canonical form for steiner designs. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 261–270. ACM, 2013. doi:10.1145/2488608.2488642.

9 Christoph Berkholz, Paul S. Bonsma, and Martin Grohe. Tight lower and upper bounds for the complexity of canonical colour refinement. *Theory Comput. Syst.*, 60(4):581–614, 2017. doi:10.1007/s00224-016-9686-0.

10 Jin-yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identifications. *Combinatorica*, 12(4):389–410, 1992. doi:10.1007/BF01305232.

11 John D. Dixon and Brian Mortimer. *Permutation groups*, volume 163 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1996. doi:10.1007/978-1-4612-0731-3.

12 Michael Elberfeld and Ken-ichi Kawarabayashi. Embedding and canonizing graphs of bounded genus in logspace. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 383–392. ACM, 2014. doi:10.1145/2591796.2591865.

13 Martin Grohe and Dániel Marx. Structure theorem and isomorphism test for graphs with excluded topological subgraphs. *SIAM J. Comput.*, 44(1):114–159, 2015. doi:10.1137/120892234.

14 Martin Grohe, Daniel Neuen, and Pascal Schweitzer. A faster isomorphism test for graphs of small degree. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 89–100. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00018.

15 Martin Grohe, Daniel Neuen, Pascal Schweitzer, and Daniel Wiebking. An improved isomorphism test for bounded-tree-width graphs. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 67:1–67:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.67.

16 Martin Grohe, Daniel Neuen, and Daniel Wiebking. Isomorphism testing for graphs excluding small minors. *CoRR*, abs/2004.07671, 2020. arXiv:2004.07671.

17 Martin Grohe and Pascal Schweitzer. Isomorphism testing for graphs of bounded rank width. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1010–1029. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.66.

18 Frank Harary. *Graph theory*. Addison-Wesley, 1991.

19 Neil Immerman and Eric Lander. Describing graphs: A first-order approach to graph canonization. In Alan L. Selman, editor, *Complexity Theory Retrospective: In Honor of Juris Hartmanis on the Occasion of His Sixtieth Birthday, July 5, 1988*, pages 59–81. Springer New York, New York, NY, 1990. doi:10.1007/978-1-4612-4478-3_5.

20 Ken-ichi Kawarabayashi. Graph isomorphism for bounded genus graphs in linear time. *CoRR*, abs/1511.02460, 2015. arXiv:1511.02460.

21 Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *J. Comput. Syst. Sci.*, 25(1):42–65, 1982. doi:10.1016/0022-0000(82)90009-5.

22 Eugene M. Luks. Hypergraph isomorphism and structural equivalence of boolean functions. In Jeffrey Scott Vitter, Lawrence L. Larmore, and Frank Thomson Leighton, editors, *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*, pages 652–658. ACM, 1999. doi:10.1145/301250.301427.

23 Eugene M. Luks and Takunari Miyazaki. Polynomial-time normalizers. *Discret. Math. Theor. Comput. Sci.*, 13(4):61–96, 2011. URL: `http://dmtcs.episciences.org/531`.

24 Brendan D. McKay. Practical graph isomorphism. *Congr. Numer.*, 30:45–87, 1981.

25 Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *J. Symb. Comput.*, 60:94–112, 2014. `doi:10.1016/j.jsc.2013.09.003`.

26 Gary L. Miller. Isomorphism of graphs which are pairwise k-separable. *Information and Control*, 56(1/2):21–33, 1983. `doi:10.1016/S0019-9958(83)80048-5`.

27 Gary L. Miller. Isomorphism of k-contractible graphs. A generalization of bounded valence and bounded genus. *Information and Control*, 56(1/2):1–20, 1983. `doi:10.1016/S0019-9958(83)80047-3`.

28 Daniel Neuen. Graph isomorphism for unit square graphs. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPIcs*, pages 70:1–70:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. `doi:10.4230/LIPIcs.ESA.2016.70`.

29 Daniel Neuen. *The Power of Algorithmic Approaches to the Graph Isomorphism Problem*. PhD thesis, RWTH Aachen University, Aachen, Germany, 2019. `doi:10.18154/RWTH-2020-00160`.

30 Daniel Neuen. Hypergraph isomorphism for groups with restricted composition factors. *CoRR*, abs/2002.06997, 2020. `arXiv:2002.06997`.

31 Ilia N. Ponomarenko. The isomorphism problem for classes of graphs closed under contraction. *Journal of Soviet Mathematics*, 55(2):1621–1643, 1991. `doi:10.1007/BF01098279`.

32 Gerhard Ringel. Das geschlecht des vollständigen paaren graphen. *Abh. Math. Semin. Univ. Hambg.*, 28(3):139–150, 1965. `doi:10.1007/BF02993245`.

33 Joseph J. Rotman. *An introduction to the theory of groups*, volume 148 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, fourth edition, 1995. `doi:10.1007/978-1-4612-4176-8`.

34 Pascal Schweitzer and Daniel Wiebking. A unifying method for the design of algorithms canonizing combinatorial objects. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1247–1258. ACM, 2019. `doi:10.1145/3313276.3316338`.

35 Ákos Seress. *Permutation group algorithms*, volume 152 of *Cambridge Tracts in Mathematics*. Cambridge University Press, Cambridge, 2003. `doi:10.1017/CBO9780511546549`.

36 Xiaorui Sun and John Wilmes. Faster canonical forms for primitive coherent configurations: Extended abstract. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 693–702. ACM, 2015. `doi:10.1145/2746539.2746617`.

37 Daniel Wiebking. Graph isomorphism in quasipolynomial time parameterized by treewidth. *CoRR*, abs/1911.11257, 2019. `arXiv:1911.11257`.