Graph Isomorphism in Quasipolynomial Time Parameterized by Treewidth

Daniel Wiebking
RWTH Aachen University, Germany
wiebking@informatik.rwth-aachen.de

Abstract

We extend Babai's quasipolynomial-time graph isomorphism test (STOC 2016) and develop a quasipolynomial-time algorithm for the multiple-coset isomorphism problem. The algorithm for the multiple-coset isomorphism problem allows to exploit graph decompositions of the given input graphs within Babai's group-theoretic framework.

We use it to develop a graph isomorphism test that runs in time $n^{\text{polylog}(k)}$ where n is the number of vertices and k is the minimum treewidth of the given graphs and polylog(k) is some polynomial in log(k). Our result generalizes Babai's quasipolynomial-time graph isomorphism test.

2012 ACM Subject Classification Mathematics of computing → Combinatorial algorithms; Mathematics of computing → Hypergraphs; Mathematics of computing → Graph algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases Graph isomorphism, canonization, treewidth, hypergraphs

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.103

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1911.11257`.

1 Introduction

The graph isomorphism problem asks for a structure preserving bijection between two given graphs G and H, i.e., a bijection $\varphi : V(G) \to V(H)$ such that $vw \in E(G)$ if and only if $\varphi(v)\varphi(w) \in E(H)$. One central open problem in theoretical computer science is the question whether the graph isomorphism problem can be solved in polynomial time. There are a few evidences that the problem might not be NP-hard. For example, NP-hardness of the problem implies a collapse of the polynomial hierarchy [32]. Moreover, NP-hardness of the graph isomorphism problem would refute the exponential time hypothesis since the problem can be decided in quasipolynomial time [1].

The research of the graph isomorphism problem started with two fundamental graph classes, i.e., the class of trees and the class of planar graphs. In 1970, Zemlyachenko gave a polynomial-time isomorphism algorithm for trees [37]. One year later, Hopcroft and Tarjan extended a result of Weinberg and designed a polynomial-time isomorphism algorithm for planar graphs [16],[34]. In 1980, Filotti, Mayer and Miller extended the polynomial-time algorithm to graphs of bounded genus [24],[10][1]. The genus is a graph parameter that measures how far away the graph is from being planar.

In Luks's pioneering work in 1982, he gave a polynomial-time isomorphism algorithm for graphs of bounded degree [22]. His group-theoretic approach laid the foundation of many other algorithms that were developed ever since. It turns out that the research in the

[1] Myrvold and Kocay pointed out an error in Filotti's techniques [26]. However, different algorithms have been given which show that the graph isomorphism problem for graphs of bounded genus is indeed decidable in polynomial time [25, 11, 17].

graph isomorphism problem for restricted graph classes was a promising approach in tackling the graph isomorphism problem in general. Shortly after Luks's result, a combinatorial partitioning lemma by Zemlyachenko was combined with Luks's framework. This resulted in an isomorphism algorithm for graphs with n vertices in general that runs in time $2^{\mathcal{O}(\sqrt{n \log n})}$ [38],[4]. This algorithm was the fastest for decades.

In 1983, the seminal work of Robertson and Seymour in graph minors started a new era of graph theory [30]. At the same time, Miller extended Luks's group-theoretic framework to hypergraphs [25]. It turned out that the study of general structures such as hypergraphs was also a promising approach in tackling the graph isomorphism problem. In 1991, Ponomarenko could in fact use Miller's hypergraph algorithm to design a polynomial-time isomorphism algorithm for graphs excluding a minor [29].

The work of Robertson and Seymour also rediscovered the notion of treewidth [8], a graph parameter that measures how far away the graph is from being a tree. The treewidth parameter was reborn and has been studied ever since. So, researchers went back to the roots and studied the isomorphism problem for graphs of bounded treewidth. In 1990, Bodlaender gave a simple isomorphism-algorithm for graphs of treewidth k with n vertices that runs in time $n^{\mathcal{O}(k)}$ [5]. However, no FPT-algorithm was known, i.e., an isomorphism algorithm with a running time of the form $f(k) \cdot n^{\mathcal{O}(1)}$. The search of a FPT-algorithm occupied researchers over years and this open problem was explicitly stated by several authors [36, 6, 18, 19, 28, 7, 9, 12]. In 2017, Lokshtanov, Pilipczuk, Pilipczuk and Saurabh finally solved this open problem and designed a FPT-algorithm for the graph isomorphism problem [21]. Their algorithm runs in time $2^{\mathcal{O}(k^5 \log k)} n^{\mathcal{O}(1)}$ where n is the number of vertices and k is the minimum treewidth of the given graphs.

At the same time, Babai made a breakthrough and designed a quasipolynomial-time algorithm for the graph isomorphism problem in general [1]. His algorithm runs in time $n^{\mathrm{polylog}(n)}$ where n is the number of vertices and $\mathrm{polylog}(n)$ is some polynomial in $\log(n)$ (according to Helfgott's analysis the function $\mathrm{polylog}(n)$ can chosen to be quadratic in $\log(n)$ [15]). To achieve this result, Babai built on Luks's group-theoretic framework, which actually solves the more general string isomorphism problem. One of the main questions is how to combine Babai's group-theoretic algorithm with the graph-theoretic techniques that have been developed. For example, it is unclear how to exploit a decomposition of the given graphs within Babai's framework since his algorithm actually processes strings rather than graphs.

Recently, Grohe, Neuen and Schweitzer were able to extend Babai's algorithm to graphs of maximum degree d and an isomorphism algorithm was developed that runs in time $n^{\mathrm{polylog}(d)}$ [13]. They suggest that their techniques might be useful also for graphs parameterized by treewidth and conjectured that the isomorphism problem for graphs of treewidth k can be decided in time $n^{\mathrm{polylog}(k)}$.

In [14], the graph-theoretic FPT-algorithm of Lokshtanov et al. was improved by using Babai's group-theoretic algorithm and the extension given by Grohe et al. as a black box. They decomposed a graph of bounded treewidth into subgraphs with particular properties. They were able to design a faster algorithm that computes the isomorphisms between these subgraphs. However, they pointed out a central problem that arises when dealing with graph decompositions: When the isomorphisms between these subgraphs are already computed, how can they be efficiently merged in order to compute the isomorphisms between the entire graphs? This problem was named as *multiple-coset isomorphism problem* and is formally defined as follows. Given two sets $J = \{\rho_1 \Delta_1^{\mathrm{Can}}, \dots, \rho_t \Delta_t^{\mathrm{Can}}\}$ and $J' = \{\rho'_1 \Delta'^{\mathrm{Can}}_1, \dots, \rho'_t \Delta'^{\mathrm{Can}}_t\}$ where $\rho_i : V \to n, \rho'_i : V' \to n$ are bijections and $\Delta_i^{\mathrm{Can}}, \Delta'^{\mathrm{Can}}_i \leq \mathrm{Sym}([n])$ are permutation groups for all $i \in [t]$, the problem is to decide whether there are bijections $\varphi : V \to V', \psi : [t] \to [t]$ such that $\Delta_i^{\mathrm{Can}} = \Delta'^{\mathrm{Can}}_{\psi(i)}$ and $\varphi \in \rho_i \Delta_i^{\mathrm{Can}}(\rho'_{\psi(i)})^{-1}$ for all $i \in [t]$. By applying the group-theoretic black box algorithms, they achieved an improved isomorphism test for graphs of treewidth k that runs in time $2^{k \cdot \mathrm{polylog}(k)} n^{\mathcal{O}(1)}$. However, for further improvements, it did not seem to be enough to use the group-theoretic algorithms as a black box only. The question of an isomorphism algorithm that runs in time $n^{\mathrm{polylog}(k)}$ remained open.

In [31], the study of the multiple-coset isomorphism problem continued. Rather than using group-theoretic algorithms as a black box, they were able to extend Luks's group-theoretic framework to the multiple-coset isomorphism problem. In order to facilitate their recursion, they introduced the class of combinatorial objects. Their class of combinatorial objects contains hypergraphs, colored graphs, relational structures, explicitly given codes and more. However, the key idea in order to handle the involved structures recursively, was to add so-called labeling cosets to their structures. By doing so, they could combine combinatorial decomposition techniques with Luks's group-theoretic framework. This led to a simply-exponential time algorithm for the multiple-coset isomorphism problem. Although the achieved running time was far away from being quasipolynomial, their result led to improvements of several algorithms. For example, it led to the currently best algorithm for the normalizer problem (a central problem in computational group theory) [35]. However, they were not able to extend also Babai's techniques to their framework and the question of a graph isomorphism algorithm running in time $n^{\mathrm{polylog}(k)}$ remained open.

Our Contribution. In this paper, we give a quasipolynomial-time algorithm for the multiple-coset isomorphism problem. This leads to an answer of the conjecture in [13] mentioned above.

▶ **Theorem** (Theorem 10). *The graph isomorphism problem can be decided in time $n^{\mathrm{polylog}(k)}$ where n is the number of vertices and k is the minimum treewidth of the input graphs.*

When $k = \mathrm{polylog}(n)$, our algorithm runs in time $n^{\mathcal{O}(\log(\log n)^c)}$ (for some constant c) and is significantly faster than Babai's algorithm and existing FPT-algorithms for graphs parameterized by treewidth.

For the present work, we exploit the fact that Babai's algorithm was recently extended to canonization [3]. A canonical labeling of a graph is a function that labels the vertices V of the graph with integers $1, \ldots, |V|$ in such a way that the labeled versions of two isomorphic graphs are equal (rather than isomorphic). The computation of canonical forms and labelings, rather than isomorphism testing, is an important task in the area of graph isomorphism and is especially useful for practical applications. Also the framework given in [31] is actually designed for the canonization problem. The present paper is based on these works and our algorithms provide canonical labelings as well. Only the algorithm given in the last section depends on the bounded-degree isomorphism algorithm of Grohe et al. for which no adequate canonization version is known.

The first necessary algorithm that we provide in our work is a simple canonization algorithm for hypergraphs.

▶ **Theorem** (Theorem 6). *Canonical labelings for hypergraphs (V, H) can be computed in time $(|V| + |H|)^{\mathrm{polylog}|V|}$.*

There is a simple argument why this algorithm is indeed necessary for our main result. It is well-known that a hypergraph $X = (V, H)$ can be encoded as a bipartite graph $G_X = (V \cup H, E)$ (the bipartite graph G_X has an edge $(v, S) \in E$, if and only if $v \in S$). It is not hard to show that the treewidth k of this bipartite graph G_X is at most $|V|$. The bipartite graph G_X uniquely encodes the hypergraph X, in particular, two hypergraphs are isomorphic if and only if their corresponding bipartite graphs are isomorphic. This means that an isomorphism algorithm for graphs of treewidth k running in time $n^{\mathrm{polylog}(k)}$ would imply an isomorphism algorithm for hypergraphs running in time $(|V|+|H|)^{\mathrm{polylog}|V|}$. However, applying Babai's algorithm to the bipartite graph would lead to a running time of $(|V|+|H|)^{\mathrm{polylog}(|V|+|H|)}$. Instead of applying Babai's algorithm to the bipartite graph directly, we decompose the hypergraph and canonize the substructures recursively. To merge the canonical labelings of all subhypergraphs, we use a canonical version of the multiple-coset isomorphism problem. However, for the hypergraph algorithm, it suffices to use Babai's algorithm as a black box only.

Our decomposition technique for hypergraphs can also be used to design a simple canonization algorithm for k-ary relations.

▶ **Theorem** (Theorem 5). *Canonical labelings for k-ary relations $R \subseteq V^k$ can be computed in time $2^{\mathrm{polylog}|V|}|R|^{\mathcal{O}(1)}$.*

The algorithm improves the currently best algorithm from [13]. As graphs can be seen as binary relations, our algorithm generalizes the quasipolynomial-time bound for graphs. The achieved running time is the best one can hope for as long as the graph isomorphism problem has no solution better than quasipolynomial time.

Our main algorithm finally solves the multiple-coset isomorphism problem. In fact, the algorithm computes canonical labelings as well.

▶ **Theorem** (Theorem 7). *Canonical labelings for a set J consisting of labeling cosets can be computed in time $(|V|+|J|)^{\mathrm{polylog}|V|}$.*

This result is actually of independent interest as it also implies a faster canonization algorithm for the entire class of combinatorial objects.

To solve this problem, the simple hypergraph canonization algorithm can be used as a subroutine in some places. However, we do not longer use Babai's and Luks's techniques as a black box only. To extend their methods, we follow the route of [31] and consider combinatorial objects that allows to combine combinatorial structures with permutation group theory. In particular, we can extend Luks's subgroup reduction and Babai's method and aggregation of local certificates to our framework. All these methods were designed for the string isomorphism problem and need non-trivial extensions when dealing with a set of labeling cosets rather than a string.

Related Work. Another extension of Babai's quasipolynomial time algorithm has been independently proposed by Daniel Neuen [27] who provided another algorithm for the isomorphism problem of hypergraphs. However, Neuen can exploit groups with restricted composition factors that are given as additional input in order to speed up his algorithm. This can be exploited in the setting of graphs of bounded Euler genus. He provides a graph isomorphism algorithm that runs in time $n^{\mathrm{polylog}(g)}$ where n is the number of vertices and g is the minimum genus of the given graphs.

On the other hand, his algorithm is not able to handle labeling cosets occurring in the combinatorial structures. In particular, his algorithm is not able to solve the multiple-coset isomorphism problem in the desired time bound, which we require for our isomorphism algorithm for graphs parameterized by treewidth. Moreover, his techniques do not provide canonical labelings.

We hope that both algorithms can be combined to give a faster isomorphism test for the large class of graphs excluding a topological subgraph. This large class of graphs includes the graphs of bounded treewidth, graphs of bounded genus, graphs of bounded degree and graphs excluding a minor. In fact, Grohe and Marx provide a structure theorem which shows that the graph classes mentioned above also characterize graphs excluding a topological subgraph. Informally, they showed that graphs excluding a topological subgraph can be decomposed into almost bounded-degree parts and minor-free parts which in turn can be decomposed into almost-embeddable parts [12]. Therefore, we hope that the improved algorithms for the isomorphism problem for bounded-degree graphs and bounded-genus graphs can be combined with our algorithm to exploit the occurring graph decomposition.

Organization of the Paper. In Section 3, we show how (sufficiently small) instances of the multiple-coset isomorphism and canonization problem can be processed with Babai's algorithm. In Section 4, we present a partitioning technique to reduce the canonization problem for k-ary relations to instances of small size in each decomposition level. In Section 5, we extend our technique to canonization of hypergraphs, which is an important subroutine used in the next section. In Section 6, we finally present our main algorithm which canonizes a set of labeling cosets and is divided into five subroutines. In the first subroutine, we extend the partitioning technique to families of partitions. The second and third subroutine extends Luks's subgroup reduction to our framework and reduces the problem to the barrier configuration characterized by a giant representation. The fourth and fifth subroutine extend Babai's method and aggregation of local certificates to our framework. In Section 7, a straightforward application of the multiple-coset isomorphism problem leads to an isomorphism algorithm that runs in time $n^{\mathrm{polylog}(k)}$ where n is the number of vertices and k is the treewidth of the given graphs.

2 Preliminaries

For an integer t, we write $[t]$ for $\{1,\dots,t\}$. For a set S and an integer k, we write $\binom{S}{k}$ for the k-element subsets of S and 2^S for the power set of S. The composition of two functions $f: V \to U$ and $g: U \to W$ is denoted by fg and is defined as the function that first applies f and then applies g.

Labeling Cosets. A *labeling* of a set V is a bijection $\rho: V \to \{1,\dots,|V|\}$. A *labeling coset* of a set V is a set of bijections Λ such that $\Lambda = \Delta\rho = \{\delta\rho \mid \delta \in \Delta\}$ for some subgroup $\Delta \le \mathrm{Sym}(V)$ and some labeling $\rho: V \to \{1,\dots,|V|\}$. We write $\mathrm{Label}(V)$ to denote the labeling coset $\mathrm{Sym}(V)\rho = \{\sigma\rho \mid \sigma \in \mathrm{Sym}(V)\}$ where ρ is an arbitrary labeling of V. Analogous to subgroups, a set $\Theta\tau$ is called a *labeling subcoset* of $\Delta\rho$, written $\Theta\tau \le \Delta\rho$, if the labeling coset $\Theta\tau$ is a subset of $\Delta\rho$.

Generating Sets. For the basic theory of handling permutation groups given by generating sets, we refer to [33]. Indeed, most algorithms are based on strong generating sets. However, given an arbitrary generating set, the Schreier-Sims algorithm is used to compute a strong generating set (of size quadratic in the degree) in polynomial time.

Hereditarily Finite Sets and Combinatorial Objects. Inductively, we define *hereditarily finite sets*, denoted by $\mathrm{HFS}(V)$, over a ground set V.

- A vertex $v \in V$ is an atom and a hereditarily finite set $v \in \mathrm{HFS}(V)$,
- a labeling coset $\Delta\rho \le \mathrm{Label}(V)$ is an atom and a hereditarily finite set $\Delta\rho \in \mathrm{HFS}(V)$,
- if $X_1,\dots,X_t \in \mathrm{HFS}(V)$, then also $\mathcal{X} = \{X_1,\dots,X_t\} \in \mathrm{HFS}(V)$ where $t \in \mathbb{N} \cup \{0\}$, and
- if $X_1,\dots,X_t \in \mathrm{HFS}(V)$, then also $\mathcal{X} = (X_1,\dots,X_t) \in \mathrm{HFS}(V)$ where $t \in \mathbb{N} \cup \{0\}$.

A *(combinatorial) object* is a pair $(V, \mathcal{X})$ consisting of a ground set V and a hereditarily finite set $\mathcal{X} \in \mathrm{HFS}(V)$. The ground set V is usually apparent from context and the combinatorial object $(V, \mathcal{X})$ is identified with the hereditarily finite set $\mathcal{X}$. The set $\mathrm{Obj}(V)$ denotes the set of all objects over V. The *transitive closure* of an object $\mathcal{X}$, denoted by $\mathrm{TC}(\mathcal{X})$, is defined as all objects that recursively occur in $\mathcal{X}$. All labeling cosets that occur in $\mathcal{X}$ are succinctly represented via generating sets. The encoding size of an object $\mathcal{X}$ can be chosen polynomial in $|\mathrm{TC}(\mathcal{X})| + |V| + t_{\max}$ where $t_{\max}$ is the maximal length of a tuple in $\mathrm{TC}(\mathcal{X})$.

Isomorphisms and Automorphisms of Objects. The image of an object $\mathcal{X} \in \mathrm{Obj}(V)$ (where V is disjoint from $\mathbb{N}$) under a bijection $\mu : V \to V'$ is an object $\mathcal{X}^\mu \in \mathrm{Obj}(V')$ that is defined as follows. Define $v^\mu := \mu(v)$ for an atom $\mathcal{X} = v$ and define $(\Delta\rho)^\mu := \mu^{-1}\Delta\rho$ for an atom $\mathcal{X} = \Delta\rho$, and inductively define $\{X_1, \ldots, X_t\}^\mu := \{X_1^\mu, \ldots, X_t^\mu\}$ and $(X_1, \ldots, X_t)^\mu := (X_1^\mu, \ldots, X_t^\mu)$.

The set of all isomorphisms from an object $\mathcal{X} \in \mathrm{Obj}(V)$ and to an object $\mathcal{X}' \in \mathrm{Obj}(V')$ is denoted by $\mathrm{Iso}(\mathcal{X}; \mathcal{X}') := \{\varphi : V \to V' \mid \mathcal{X}^\varphi = \mathcal{X}'\}$. The set of all automorphisms of an object $\mathcal{X}$ is denoted by $\mathrm{Aut}(\mathcal{X}) := \mathrm{Iso}(\mathcal{X}; \mathcal{X})$.

▶ **Definition 1** ([31]). *Let $\mathcal{C}$ be an isomorphisms-closed class of (unordered) objects, i.e., for all $\mathcal{X} \in \mathcal{C}$ over a set V and all bijections $\varphi : V \to V'$ it holds that $\mathcal{X}^\varphi \in \mathcal{C}$. A* canonical labeling function CL *is a function that assigns each object in $\mathcal{C}$ a labeling coset $\mathrm{CL}(\mathcal{X}) = \Lambda \le \mathrm{Label}(V)$ such that:*
(CL1) $\mathrm{CL}(\mathcal{X}) = \varphi\,\mathrm{CL}(\mathcal{X}^\varphi)$ *for all $\varphi \in \mathrm{Iso}(V; V')$ (the set of bijections from V to V'), and*
(CL2) $\mathrm{CL}(\mathcal{X}) = \mathrm{Aut}(\mathcal{X})\pi$ *for some (and thus for all) $\pi \in \mathrm{CL}(\mathcal{X})$.*
In this case, the labeling coset Λ is also called a canonical labeling *for $\mathcal{X}$.*

3 Handling Small Objects via String Canonization

Next, we define the central problem of this paper which is introduced in [14],[31]. This problem is a canonical version of the multiple-coset isomorphism problem.

▶ **Problem 2.** *Compute a function $\mathrm{CL}_{\mathrm{Set}}$ with the following properties:*

Input	$J \in \mathrm{Obj}(V)$ *where* $J = \{\Delta_1\rho_1, \ldots, \Delta_t\rho_t\}$, $\Delta_i\rho_i \le \mathrm{Label}(V), i \in [t]$ *and V is an (unordered) set.*
Output	*A labeling coset* $\mathrm{CL}_{\mathrm{Set}}(J) = \Lambda \le \mathrm{Label}(V)$ *such that:*
(CL1)	$\mathrm{CL}_{\mathrm{Set}}(J) = \varphi\,\mathrm{CL}_{\mathrm{Set}}(J^\varphi)$ *for all* $\varphi \in \mathrm{Iso}(V; V')$.
(CL2)	$\mathrm{CL}_{\mathrm{Set}}(J) = \mathrm{Aut}(J)\pi$ *for some (and thus for all)* $\pi \in \Lambda$.

Following the definition of the automorphism group for objects in general, we have that $\mathrm{Aut}(J) = \{\sigma \in \mathrm{Sym}(V) \mid \exists\psi \in \mathrm{Sym}(t)\forall i \in [t] : \sigma^{-1}\Delta_i\rho_i = \Delta_{\psi(i)}\rho_{\psi(i)}\}$.

To clarify the complexity status of Problem 2, it is important to note that the problem is actually polynomial-time equivalent to the string canonization problem. The string canonization problem in turn can be solved in quasipolynomial-time with Babai's algorithm [3]. However, the reduction increases the permutation domain V by a factor $|J|$, which leads to a running time of $2^{\mathrm{polylog}(|V|+|J|)}$ as stated in the following lemma.

▶ **Lemma 3.** *Canonical labelings for sets J can be computed in time $2^{\mathrm{polylog}(|V|+|J|)}$.*

The main task in this work is to remove the dependency of $|J|$ in the exponent. The main algorithm (Theorem 7) solves Problem 2 in a running time of $(|V| + |J|)^{\mathrm{polylog}|V|}$ and is presented in Section 6. This improvement will finally lead to an improved algorithm for the graph isomorphism problem from $n^{\mathrm{polylog}(n)}$ to $n^{\mathrm{polylog}(k)}$ where n is the number of vertices and k is the minimum treewidth of the input graphs. However, Lemma 3 is still used in our algorithms. Especially, when $|J|$ is bounded by some quasipolynomial in $|V|$, the algorithm from Lemma 3 already runs in the desired time bound.

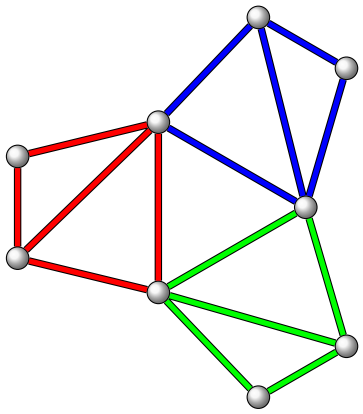

Figure 1 We see a graph G decomposed into 3 isomorphic subgraphs $H_1, H_2, H_3 \subseteq G$ shown in distinct colors.

The Intuition Behind this Central Problem. We explain why Problem 2 is the central problem when dealing with graph decompositions. We want to keep this subsection as simple as possible and do not want to introduce tree decompositions yet. For our purpose, we consider a simplified formulation of a graph decomposition. In this subsection, a graph decomposition of a graph $G = (V, E)$ is a family of subgraphs $\{H_i\}_{i\in[t]}$ that covers the edges of the entire graph, i.e., $E(G) = E(H_1) \cup \ldots \cup E(H_t)$. We say that a graph decomposition is defined in an isomorphism-invariant way if for two isomorphic graphs G, G' the decompositions $\{H_i\}_{i\in[t]}, \{H_i'\}_{i\in[t]}$ are defined in such a way that each isomorphism $\varphi \in \mathrm{Iso}(G; G')$ also maps each subgraph H_i of the decomposition of G to a subgraph H_j' of the decomposition of G'. In particular, such a decomposition has to be invariant under automorphisms of the graph. A prime example of such a isomorphism-invariant decomposition in the setting of bounded-treewidth graphs is the decomposition into clique-separator-free parts. The clique-separator decomposition goes back to Leimer [20] and is also used in our final isomorphism algorithm.

Assume we have given a graph G for which we can construct a graph decomposition $\{H_i\}_{i\in[t]}$ in an isomorphism-invariant way and our task is the computation of a canonical labeling for G. A priori, it is unclear how to exploit our graph decomposition. In a first step, we could compute canonical labelings $\Delta_i\rho_i := \mathrm{CL}(H_i)$ for each subgraph H_i recursively. The central question is how to merge these labeling cosets $\Delta_i\rho_i$ for H_i in order to obtain a canonical labeling $\Delta\rho$ for the entire graph G.

The easy case occurs when all subgraphs H_i, H_j are pairwise non-isomorphic. In this case, the subgraphs cannot be mapped to each other and indeed $\mathrm{Aut}(G) = \mathrm{Aut}(H_1) \cap \ldots \cap \mathrm{Aut}(H_t)$. Therefore, the computation of $\Delta\rho$ reduces to a canonical intersection-problem. In fact, Babai's quasipolynomial-time algorithm [3] can be used to intersect labeling cosets canonically.

We consider the second extreme case in which all subgraphs H_i, H_j are pairwise isomorphic, see Figure 1.

In such a case, we have that $\mathrm{Aut}(G) = \{\sigma \in \mathrm{Sym}(V) \mid \exists \psi(t) \forall i \in [t] : \sigma \in \mathrm{Iso}(H_i; H_{\psi(i)})\}$. Equivalently, we have that $\mathrm{Aut}(G) = \mathrm{Aut}(\{\Delta_1\rho_1, \ldots, \Delta_t\rho_t\})$. Therefore, by the definition of Problem 2, the canonical labeling $\Delta\rho := \mathrm{CL}_{\mathrm{Set}}(\{\Delta_1\rho_1, \ldots, \Delta_t\rho_t\})$ defines a canonical labeling for the entire graph G.

Alternatively, one can use the following lemma which intuitively says that for the purpose of canonization the subgraphs H_i can be replaced with their labeling cosets $\Delta_i\rho_i$ while preserving all symmetry information. Formally, it says the following.

▶ **Lemma 4** ([31], Object Replacement Lemma). *Let $\mathcal{X} = \{X_1, \dots, X_t\}$ be an object and let* CL *and* $\mathrm{CL}_{\mathrm{Set}}$ *be canonical labeling functions. Define $\mathcal{X}^{\mathrm{Set}} := \{\Delta_1\rho_1, \dots, \Delta_t\rho_t\}$ where $\Delta_i\rho_i := \mathrm{CL}(X_i)$ is a canonical labeling for $X_i \in \mathcal{X}$. Assume that $X_i, X_j \in \mathcal{X}$ are pairwise isomorphic. Then, $\mathrm{CL}_{\mathrm{Object}}(\mathcal{X}) := \mathrm{CL}_{\mathrm{Set}}(\mathcal{X}^{\mathrm{Set}})$ defines a canonical labeling for $\mathcal{X}$.*

Roughly speaking, Problem 2 can be seen as the task of merging the given labeling cosets. The mixed case in which some (but not all) subgraphs H_i, H_j are isomorphic can be handled by a mixture of the above cases.

In Section 7, we apply this problem to graphs G with n vertices of treewidth k. By exploiting that the subgraphs in our application can only intersect in cliques of size at most k, we are able to restrict our attention to vertex sets of size $|V| \le k$ (with at most $|J| \le n$ labeling cosets). This finally leads to the desired running time of $n^{\mathrm{polylog}(k)}$.

Application to Combinatorial Objects. A second application of Problem 2 is the canonization framework for combinatorial objects in general given in [31]. In fact, our algorithms build on this canonization framework as it allows a recursive approach to solve the problem. Our improved running time for Problem 2 then implies an improved canonization algorithm for combinatorial objects in general that runs in time $n^{\mathrm{polylog}|V|}$ (Corollary 8).

4 Canonization of k-ary Relations

In this section, we provide an algorithm for canonical labeling of k-ary relations $R \subseteq V^k$. As graphs can be seen as binary relations, this problem clearly generalizes the graph canonization problem. One way to canonize k-ary relations is by using a well-known reduction to the graph canonization problem [23]. However, this approach leads to a running time that is quasipolynomial in $|V| + |R|$. In this section, we will give a polynomial-time reduction to the canonization problem for objects that are of input size polynomial in $|V|$ (which does not depend on $|R|$). With this reduction, we obtain an improved algorithm that runs in time $2^{\mathrm{polylog}|V|}|R|^{\mathcal{O}(1)}$. Our bound improves the currently best algorithm from [13]. Moreover, our time bound is also optimal (when measured in $|V|$ and $|R|$) as long as the graph isomorphism problem can not be solved faster than quasipolynomial time.

A *partition* of a set $\mathcal{X} \in \mathrm{Obj}(V)$ is a set $\mathcal{P} = \{P_1, \dots, P_p\}$ such that $\mathcal{X} = P_1 \cup \dots \cup P_p$ where $\emptyset \neq P_i \subseteq \mathcal{X}$ for all $P_i \in \mathcal{P}$. We suggest a general technique for exploiting partitions.

The Partitioning Technique. In this setting, we assume that we are given some object $\mathcal{X} \in \mathrm{Obj}(V)$ for which we can construct a partition $\mathcal{P} = \{P_1, \dots, P_p\}$ in an isomorphism-invariant way such that $2 \le |\mathcal{P}| \le 2^{\mathrm{polylog}|V|}$. The goal is the computation of a canonical labeling for $\mathcal{X}$ by using an efficient recursion.

For example, $\mathcal{X} = R \subseteq V^k$ might be a k-ary relation for which we can easily construct a partition in an isomorphism-invariant way, as seen next. Assume $|R| \ge 2$ (otherwise the canonization problem is easy to solve) and let r be the first position in which R differs, i.e., the smallest $r \in [k]$ such that there are $(x_1, \dots, x_k), (y_1, \dots, y_k) \in R$ with $x_r \neq y_r$. Then, we partition $R = P_1 \cup \dots \cup P_p$ by saying that two tuples $(x_1, \dots, x_k), (y_1, \dots, y_k)$ are in the same part P_i if and only if $x_r = y_r$. This gives a non-trivial partition $\mathcal{P} = \{P_1, \dots, P_p\}$ with $2 \le |\mathcal{P}| \le |V| \le 2^{\mathrm{polylog}|V|}$ which is preserved under automorphisms and isomorphisms.

Using recursion, we compute a canonical labeling $\Delta_i\rho_i$ for each part $P_i \subseteq \mathcal{X}$ recursively (assumed that we can define a partition for each part again). In our example, $P_i \subseteq R$ is a subrelation and therefore we can apply our approach recursively.

So far, we computed canonical labelings for each part $P_i \subseteq \mathcal{X}$ independently. The main idea is to use our central problem (Problem 2) to merge all these labeling cosets. Let us restrict our attention to the case in which the parts $P_i, P_j \in \mathcal{P}$ are pairwise isomorphic. In this case, we define the set $\mathcal{P}^{\text{Set}} := \{\Delta_i \rho_i \mid P_i \in \mathcal{P}\}$ consisting of the canonical labelings $\Delta_i \rho_i$ for each part. Moreover, by object replacement (Lemma 4), a canonical labeling for $\mathcal{P}^{\text{Set}}$ defines a canonical labeling for $\mathcal{P}$ as well. A canonical labeling for $\mathcal{P}$ in turn defines a canonical labeling for $\mathcal{X}$ since we assume the partition to be defined in an isomorphism-invariant way. Therefore, it is indeed true that a canonical labeling for $\mathcal{P}^{\text{Set}}$ would define a canonical labeling for $\mathcal{X}$. For this reason, we can use the algorithm from Lemma 3 to compute a canonical labeling for $\mathcal{P}^{\text{Set}}$. Intuitively, this algorithm merges all the labeling cosets in $\mathcal{P}^{\text{Set}}$ into one single canonical labeling. In our example, this single labeling coset is a canonical labeling for the relation R. The algorithm Lemma 3 runs in the desired time bound since $|\mathcal{P}^{\text{Set}}| = |\mathcal{P}| \leq 2^{\text{polylog}|V|}$ is bounded by some quasipolynomial.

Let us consider the number of recursive calls $R(\mathcal{X})$ of this approach for a given object $\mathcal{X}$. Since we recurse on each part $P_i \in \mathcal{P}$, we have a recurrence of $R(\mathcal{X}) = 1 + \sum_{P_i \in \mathcal{P}} R(P_i)$ leading to at most $|\mathcal{X}|^{\mathcal{O}(1)}$ recursive calls. The running time for one single recursive call is bounded by $2^{\text{polylog}|V|}$. For this reason, the total running time is bounded by $2^{\text{polylog}|V|}|\mathcal{X}|^{\mathcal{O}(1)}$.

▶ **Theorem 5.** *Canonical labelings for k-ary relations $R \subseteq V^k$ can be computed in time $2^{\text{polylog}|V|}|R|^{\mathcal{O}(1)}$.*

5 Canonization of Hypergraphs

In this section, we provide an algorithm for canonical labeling of hypergraphs (V, H) where $H \subseteq 2^V$.

We want to extend the previous partitioning technique to hypergraphs. However, for hypergraphs a non-trivial isomorphism-invariant partition $H = H_1 \uplus \ldots \uplus H_s$ of the edge set does not always exist, e.g., the hypergraph $(V, \{S \subseteq V \mid |S| = 2\})$ does not have a non-trivial partition of the edge set that is preserved under automorphisms. Therefore, we can not apply the partitioning technique to this setting. For this reason, we introduce a generalized technique in order to solve this problem. This generalized technique results in a slightly weaker time bound of $(|V| + |H|)^{\text{polylog}|V|}$ (where the dependency on $|H|$ is not polynomial). Indeed, it is an open problem whether the running time for the hypergraph isomorphism problem can be improved to $2^{\text{polylog}|V|} \cdot |H|^{\mathcal{O}(1)}$ [2].

A *cover* of a set $\mathcal{X} \in \text{Obj}(V)$ is a set $\mathcal{C} = \{C_1, \ldots, C_c\}$ such that $\mathcal{X} = C_1 \cup \ldots \cup C_c$ where $\emptyset \neq C_i \subseteq \mathcal{X}$ for all $C_i \in \mathcal{C}$. In contrast to a partition, the sets C_i, C_j are not necessarily disjoint for $i \neq j$. A cover $\mathcal{C}$ of $\mathcal{X}$ is called *sparse* if $|C_i| \leq \frac{1}{2}|\mathcal{X}|$ for all $C_i \in \mathcal{C}$. Extending the partitioning technique, we suggest a technique to handle covers.

The Covering Technique. In this setting, we assume that we have given some object $\mathcal{X} \in \text{Obj}(V)$ for which we can define a cover $\mathcal{C} = \{C_1, \ldots, C_c\}$ in an isomorphism-invariant way. Also here, we assume that $2 \leq |\mathcal{C}| \leq 2^{\text{polylog}|V|}$. The goal is the computation of a canonical labeling of $\mathcal{X}$ using an efficient recursion.

For example, $\mathcal{X} = H$ is a hypergraph for which we can easily define a cover in an isomorphism-invariant way, as seen next. We assume that $|H| \geq 2$ (otherwise the canonization problem is easy to solve) and that $\emptyset \notin H$ (otherwise we remove the empty set from H). Then, we cover $H = \bigcup_{v \in V} C_v$ by setting $C_v := \{S \in H \mid v \in S\}$ and define a cover $\mathcal{C} := \{C_v \mid v \in V\}$. Since each hyperedge $S \in H$ contains at least one vertex $v \in V$, each hyperedge S is contained in at least one set $C_v \in \mathcal{C}$. Moreover, the cover $\mathcal{C}$ is preserved under automorphisms and isomorphisms.

First, we reduce to the setting in which $\mathcal{C}$ is a sparse cover of $\mathcal{X}$. This can be done as follows. We define $C_i^* := C_i$ if $|C_i| \leq \frac{1}{2}|\mathcal{X}|$ and we define $C_i^* := \mathcal{X} \setminus C_i$ if $|C_i| > \frac{1}{2}|\mathcal{X}|$. By definition, we ensured that $|C_i^*| \leq \frac{1}{2}|\mathcal{X}|$ for all $i \in [c]$. Let $\mathcal{X}^* := \bigcup_{i\in[c]} C_i^*$. Next, we consider two cases.

If $\mathcal{X}^* \subsetneq \mathcal{X}$, then we have found a non-trivial partition $\mathcal{X} = \mathcal{X}^* \uplus \mathcal{X}^\circ$ where $\mathcal{X}^\circ := \mathcal{X} \setminus \mathcal{X}^*$. In the hypergraph example, we would have a non-trivial partition of the hyperedges defined in an isomorphism-invariant way (which can be exploited very easily). We proceed analogously as in the partitioning technique explained in Section 4.

Otherwise, if $\mathcal{X}^* = \mathcal{X}$, then $\mathcal{C}^* := \{C_1^*, \ldots, C_c^*\}$ is also a cover of $\mathcal{X}$. But more importantly, the cover $\mathcal{C}^*$ is indeed sparse. In the case of a sparse cover, we also proceed analogously as in the partition technique explained in Section 4. However, the key difference of the covering technique compared to the partitioning technique lies in the recurrence for the number of recursive calls since the sets $C_i^*, C_j^* \in \mathcal{C}^*$ are not necessarily pairwise disjoint. The recurrence we have is $R(\mathcal{X}) = 1 + \sum_{C_i^* \in \mathcal{C}^*} R(C_i^*)$. By using that $|\mathcal{C}^*| = |\mathcal{C}| \leq 2^{\operatorname{polylog}|V|}$ and that $|C_i^*| \leq \frac{1}{2}|\mathcal{X}|$, we obtain at most $|\mathcal{X}|^{\operatorname{polylog}|V|}$ recursive calls. This is exactly the reason why the algorithm for relations is faster than the algorithm for hypergraphs.

▶ **Theorem 6.** *Canonical labelings for hypergraphs* (V, H) *can be computed in time* $(|V| + |H|)^{\operatorname{polylog}|V|}$.

6 Canonization of Sets and Objects

Our main theorem provides an algorithm that canonizes a set $J = \{\Delta_1\rho_1, \ldots, \Delta_t\rho_t\}$ consisting of labelings cosets $\Delta_i\rho_i \leq \operatorname{Label}(V), i \in [t]$.

▶ **Theorem 7.** *A function* $\mathrm{CL}_{\mathrm{Set}}$ *solving Problem 2 can be computed in time* $(|V|+|J|)^{\operatorname{polylog}|V|}$.

▶ **Corollary 8.** *Canonical labelings for combinatorial objects can be computed in time* $n^{\operatorname{polylog}|V|}$ *where* n *is the input size and* V *is the ground set of the object.*

Proof Outline. For the purpose of recursion, our main algorithm $\mathrm{CL}_{\mathrm{Set}}$ needs some additional input parameters. The input of the main algorithm is a tuple $(J, A, \Delta^{\mathrm{Can}}, g^{\mathrm{Can}})$ consisting of the following input parameters.

- J is a set consisting of labeling cosets,
- $A \subseteq V$ is a subset which is Δ_i-invariant for all $\Delta_i\rho_i \in J$. We require that Property (A) holds: $(\Delta_i\rho_i)|_{V\setminus A} = (\Delta_j\rho_j)|_{V\setminus A}$ for all $\Delta_i\rho_i, \Delta_j\rho_j \in J$ (initially, we set $A := V$),
- $\Delta^{\mathrm{Can}} \leq \operatorname{Sym}(V^{\mathrm{Can}})$ is a group over the linearly ordered set $V^{\mathrm{Can}} = \{1, \ldots, |V|\}$. We require that for all $\Delta_i\rho_i \in J$ it holds that $\rho_i^{-1}\Delta_i\rho_i = \Delta^{\mathrm{Can}}$ (if this would not be the case, in can be shown that J can be partitioned such that progress can be measured), and
- $g^{\mathrm{Can}} : \Delta^{\mathrm{Can}} \to \operatorname{Sym}(W^{\mathrm{Can}})$ is a giant representation where $W^{\mathrm{Can}} = \{1, \ldots, |W^{\mathrm{Can}}|\}$ is a linearly ordered set (a homomorphism $h : \Delta \to \operatorname{Sym}(W)$ is called a *giant representation* if the image of Δ under h is a giant, i.e., $\operatorname{Alt}(W) \leq h(\Delta) \leq \operatorname{Sym}(W)$). It is allowed that g^{Can} is undefined ($g^{\mathrm{Can}} = \bot$).

The Subgroup Recursion. First of all we design a subgroup reduction that, given a tuple $(J, A, \Delta^{\mathrm{Can}}, \bot)$ and a subgroup $\Psi^{\mathrm{Can}} \leq \Delta^{\mathrm{Can}}$, reduces the canonical labeling problem of $(J, A, \Delta^{\mathrm{Can}}, \bot)$ to s-many instances $(\widehat{J}_i, A, \Psi^{\mathrm{Can}}, \bot)$ where $|\widehat{J}_i| \leq |J|$. Here, s corresponds to the index of Ψ^{Can} in Δ^{Can}. In contrast to Luks's subgroup reduction, the present reduction splits all labeling cosets in J simultaneously. We describe the idea of this algorithm.

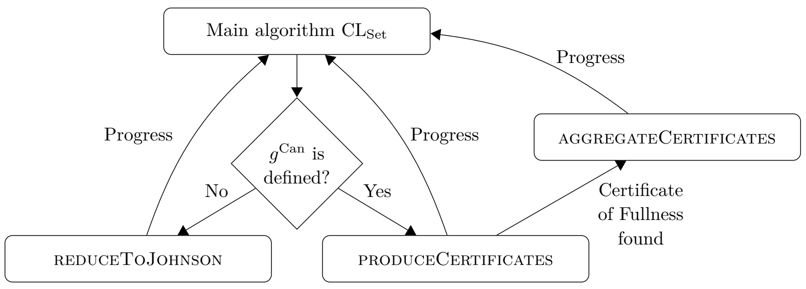

Figure 2 Flowchart of the algorithm for Theorem 7.

We consider the decomposition into left cosets of $\Delta^{\text{Can}} = \bigcup_{\ell \in [s]} \delta_\ell^{\text{Can}} \Psi^{\text{Can}}$ and define $\widehat{J} := \{\rho_i \delta_\ell^{\text{Can}} \Psi^{\text{Can}} \mid i \in [t], \ell \in [s]\}$. Surprisingly, we can show that $\text{Aut}(\widehat{J}) = \text{Aut}(J)$. This means that a canonical labeling for $\widehat{J}$ defines a canonical labeling for J as well and vice versa. Therefore, the first idea that comes to mind would be a recursion on the instance $(\widehat{J}, A, \Psi^{\text{Can}}, \bot)$. However, there are two problems when recursing on $\widehat{J}$. First, the instance $\widehat{J}$ does not necessarily preserve Property (A) that we have for the subset $A \subseteq V$ (this requirement is important for the subroutines that follow).

Second, it holds that $|\widehat{J}| > |J|$ (assumed that $\Psi^{\text{Can}} < \Delta^{\text{Can}}$ is a proper subgroup). Also this blow-up in the instance size would not lead to the desired recursion. However, we are able to construct a decomposition of $\widehat{J} = \widehat{J}_1 \cup \ldots \cup \widehat{J}_r$ such that $r \leq s$ and $|\widehat{J}_i| \leq |J|$ and such that Property (A) holds for each instance $(\widehat{J}_i, A, \Psi^{\text{Can}}, \bot)$.

The Johnson Reduction. We design an algorithm that, given an instance $(J, A, \Delta^{\text{Can}}, \bot)$, either finds a giant representation $g^{\text{Can}} : \Delta^{\text{Can}} \to \text{Sym}(W^{\text{Can}})$ or reduces the canonical labeling problem of $(J, A, \Delta^{\text{Can}}, \bot)$ to instances that are smaller (according to some function that measures progress).

First of all, we want to reduce to the case in which all $\Delta_i \leq \text{Sym}(V)$ are transitive on $A \subseteq V$. To achieve transitivity, Babai's algorithm uses Luks's idea of orbit-by-orbit processing. However, the orbit-by-orbit recursion is a tool that is developed for strings and needs a non-trivial adaption when dealing with a set of labeling cosets J. To achieve transitivity, the present algorithm uses an extension of the orbit-by-orbit recursion that was developed in [31]. In the transitive case, we follow Babai's idea. First, we define a block system $\mathcal{B}^{\text{Can}}$ on which Δ^{Can} acts primitively. If the primitive group acting on $\mathcal{B}^{\text{Can}}$ is small, we use the subgroup reduction to reduce to a subgroup $\Psi^{\text{Can}} \leq \Delta^{\text{Can}}$ that is defined as the kernel of that action. In case that the primitive group is large, we use Cameron's classification of large primitive groups which implies that the primitive group is a Cameron group. We reduce the Cameron group to a Johnson group by using the subgroup reduction again. The Johnson group (acting on subsets of a set W^{Can}) in turn can be used to define a giant representation $g^{\text{Can}} : \Delta^{\text{Can}} \to \text{Sym}(W^{\text{Can}})$.

▶ **Definition 9** (Certificates of Fullness). *A group* $G \leq \text{Sym}(V)$ *is called* certificate of fullness *for an instance* $(J, A, \Delta^{\text{Can}}, g^{\text{Can}})$ *if*

1. $G \leq \text{Aut}(J)$,
2. $G^{\text{Can}} := G^{\rho_i} \leq \Delta^{\text{Can}}$ *does not depend on the choice of* $\Delta_i \rho_i \in J$, *and*
3. $g^{\text{Can}} : G^{\text{Can}} \to \text{Sym}(W^{\text{Can}})$ *is still a giant representation.*

The Certificate Producing Algorithm. We design an algorithm that, given an instance $(J, A, \Delta^{\text{Can}}, g^{\text{Can}})$ (where g^{Can} is defined), either finds a certificate of fullness or makes progress (according to some function that measures progress).

The algorithm picks a subset $T^{\mathrm{Can}} \subseteq W^{\mathrm{Can}}$ of logarithmic size. We call this set T^{Can} a *canonical test set.* Next, we define the group $\Delta_T^{\mathrm{Can}} \leq \Delta^{\mathrm{Can}}$ which stabilizes T^{Can} in the image under g^{Can}. By doing so, we can define a giant representation $g_T^{\mathrm{Can}} : \Delta_T^{\mathrm{Can}} \to \mathrm{Sym}(T^{\mathrm{Can}})$. We say that $v^{\mathrm{Can}} \in V^{\mathrm{Can}}$ is *affected* by g_T^{Can} if g_T^{Can} is not a giant representation when restricted to $(\Delta_T^{\mathrm{Can}})_{(v^{\mathrm{Can}})}$. Let $S^{\mathrm{Can}}, U^{\mathrm{Can}} \subseteq V^{\mathrm{Can}}$ be set of elements affected and unaffected by g_T^{Can}, respectively. We have a technical difference in our algorithm in contrast to Babai's method. In Babai's method of local certificates, he processes a giant representation $g : \Delta \to \mathrm{Sym}(W)$ and considers multiple test sets $T \subseteq W$ (one test set for each subset of logarithmic size). In our framework, we define the giant representation for a group Δ^{Can} over a linearly ordered set V^{Can}. This allows us to choose one single (canonical) test set $T^{\mathrm{Can}} \subseteq W^{\mathrm{Can}}$ only. Here, canonical means that the subset is chosen minimal with respect to the ordering of the natural numbers. However, when we translate the ordered structures V^{Can} to unordered structures over V, we implicitly consider multiple test sets and giant representations. More precise, by applying inverses of labelings in $\Delta_i\rho_i \in J$ to the ordered group $\Delta_T^{\mathrm{Can}} \leq \mathrm{Sym}(V^{\mathrm{Can}})$, we obtain a set of groups over V, i.e., $\{\lambda_i \Delta_T^{\mathrm{Can}} \lambda_i^{-1} \mid \lambda_i \in \Delta_i\rho_i\}$. Similarly, we can define a set of giant representations $\{(g_T^{\mathrm{Can}})^{\lambda_i^{-1}} \mid \lambda_i \in \Delta_i\rho_i\}$ (where $(g_T^{\mathrm{Can}})^{\lambda_i^{-1}}(\delta_i) := g_T^{\mathrm{Can}}(\lambda_i^{-1}\delta_i\lambda_i)$ for $\delta_i \in \Delta_i$) and a set of affected points $H_i := \{S \subseteq V \mid S^{\lambda_i} = S^{\mathrm{Can}}$ for some $\lambda_i \in \Delta_i\rho_i\}$. Therefore, when dealing over unordered structures, we need to consider multiple groups and homomorphisms. It becomes even more complex, since we are dealing with a set J consisting of labeling cosets rather than one single group only. In fact, we obtain a set of affected point sets H_i for each labeling coset $\Delta_i\rho_i \in J$. However, it turns out that the hardest case occurs when $H_i = H_j$ for all $\Delta_i\rho_i, \Delta_j\rho_j \in J$. Roughly speaking, we will apply the following strategy.

We restrict each labeling coset in J to some set of affected points $S \in H_i$ and define a set of local restrictions J_S^* that ignore the vertices outside S. The precise definition of J_S^* is given in our algorithm. Intuitively, the algorithms tries to analyze the labeling cosets locally.

Case 1: The local restrictions J_S^* are pairwise distinct. In this case, we canonize the local restrictions J_S^* recursively. Observe that a canonical labeling $\Delta\rho$ for J_S^* does not necessarily define a canonical labeling for J. However, we can define a function $\alpha : J_S^* \to J$ that assigns each local restriction its corresponding labeling coset $\Delta_i\rho_i \in J$. This function is well-defined since we assumed the local restrictions to be pairwise distinct. Now, each automorphism in $\mathrm{Aut}(J_S^*)$ induces a permutation of J_S^* which in turn induces a permutation of J. We are able to use the permutations on J to canonize the set J efficiently (without even applying further recursive calls).

Case 2: Some local restrictions in J_S^* are pairwise different and some local restrictions in J_S^* are pairwise equal. In this case, we can define a non-trivial partition of J in the following way. We say that two labeling cosets $\Delta_i\rho_i, \Delta_j\rho_j$ are in the same part, if and only if the corresponding local restrictions in J_S^* coincide. Actually, this leads to a family of partitions since we obtain one partition for each choice of an affected set $S \in H_i$. We exploit this partition family by using an extension of the partitioning techniques from Sections 4 and 5.

Case 3: The local restrictions J_S^* are pairwise equal. In this case, it is possible to find automorphisms $G_S \leq \mathrm{Sym}(V)$ of J which fix the unaffected points $V \setminus S$. In fact, we can find such automorphisms for all choices of $S \in H_i$, otherwise we are in a situation of a previous case. Finally, we consider the group of automorphisms $G \leq \mathrm{Aut}(J)$ generated by all G_S for $S \in H_i$. We can show that G is indeed a certificate of fullness.

The Certificate Aggregation. We finally design an algorithm that, given an instance $(J, A, \Delta^{\mathrm{Can}}, g^{\mathrm{Can}})$ and a certificate of fullness $G \leq \mathrm{Sym}(V)$ makes progress (according to some function that measures progress).

Let us consider the less technical case in which $g^{\mathrm{Can}}(G^{\mathrm{Can}})$ is the symmetric group (rather than the alternating group). In this case, it holds that $G^{\mathrm{Can}}\Psi^{\mathrm{Can}} = \Delta^{\mathrm{Can}}$ where Ψ^{Can} is the kernel of g^{Can}. Similarly to the subgroup recursion, we consider the decomposition of $\Delta^{\mathrm{Can}} = \bigcup_{\ell\in[s]} \delta_\ell^{\mathrm{Can}}\Psi^{\mathrm{Can}}$ into left cosets of the kernel and define $\widehat{J} := \{\rho_i\delta_\ell^{\mathrm{Can}}\Psi^{\mathrm{Can}} \mid i \in [t], \ell \in [s]\}$. Again, we have $\mathrm{Aut}(\widehat{J}) = \mathrm{Aut}(J)$. The key observation is that G is transitive on $\widehat{J}$ since $(\rho_i\delta_\ell^{\mathrm{Can}}\Psi^{\mathrm{Can}})^{g^{-1}} = \rho_i g^{\rho_i}\delta_\ell^{\mathrm{Can}}\Psi^{\mathrm{Can}}$ for all $g \in G$ and $G^{\mathrm{Can}}\Psi^{\mathrm{Can}} = \Delta^{\mathrm{Can}}$.

First, consider an easy case in which $J = \{\Delta_1\rho_1\}$ consists of one single labeling coset. In this case, we have a set of automorphisms G acting transitively on the subcosets $\widehat{J} = \{\rho_1\delta_\ell^{\mathrm{Can}}\Psi^{\mathrm{Can}} \mid \ell \in [s]\}$. Moreover, each subcoset satisfies $\mathrm{Aut}(\rho_1\delta_\ell^{\mathrm{Can}}\Psi^{\mathrm{Can}}) \leq \mathrm{Aut}(J)$ and can be seen as an individualization of J. This means, we can choose (arbitrarily) a subcoset $\rho_1\delta_\ell^{\mathrm{Can}}\Psi^{\mathrm{Can}} \leq \Delta_1\rho_1$ and recurse on that. Since the automorphisms in G can map each subcoset to each other subcoset it does not matter which subcoset we choose. By recursing on one single subcoset only, we can measure significant progress. At the end, we return $G\widehat{\Lambda}$ where $\widehat{\Lambda}$ is a canonical labeling for the (arbitrarily) chosen subcoset and G is the group of automorphisms (acting transitively on the set of all subcosets).

However, the situation becomes more difficult when dealing with more labeling cosets $J = \{\Delta_1\rho_1, \ldots, \Delta_t\rho_t\}$ for $t \geq 2$. The first idea that comes to mind is the following generalization. We choose (arbitrarily) some $\ell \in [s]$ and define the set of subcosets $\widehat{J}_\ell := \{\rho_i\delta_\ell^{\mathrm{Can}}\Psi^{\mathrm{Can}} \mid i \in [t]\} \subseteq \widehat{J}$. The set $\widehat{J}_\ell$ contains exactly one subcoset $\rho_i\delta_\ell^{\mathrm{Can}}\Psi^{\mathrm{Can}} \leq \Delta_i\rho_i$ of each $\Delta_i\rho_i \in J$. However, the partition $\widehat{\mathcal{J}} := \{\widehat{J}_\ell \mid \ell \in [s]\}$ might not be G-invariant and G might not be transitive on it. In fact, we are able to find a suitable partition $\widehat{\mathcal{J}} := \{\widehat{J}_1, \ldots, \widehat{J}_r\}$ of the subcosets $\widehat{J}$ on which G is transitive.

7 Isomorphism of Graphs Parameterized by Treewidth

▶ **Theorem 10.** *Let G_1, G_2 be two connected graphs. There is an algorithm that, given a pair (G_1, G_2), computes the set of isomorphisms $\mathrm{Iso}(G_1; G_2)$ in time $|V(G_1)|^{\mathrm{polylog}(\mathrm{tw}\, G_1)}$.*

Proof Outline. We follow the graph decomposition approach from [14] building on [21]. It was shown that graphs of treewidth k can be decomposed in an isomorphism-invariant way into parts that have restrictions on their automorphism groups. More precisely, the automorphism group of each part has composition-width at most k after fixing one vertex (the composition-width of a group Δ is the smallest integer k such that all composition factors of Δ are isomorphic to a subgroup of $\mathrm{Sym}(k)$). This fact allows us to use the bounded-degree graph isomorphism algorithm given in [13] to compute the isomorphisms between the parts. Finally, we use our main theorem (Theorem 7) to merge the isomorphisms.

Since Grohe, Neuen and Schweitzer provide an isomorphism algorithm, rather than a canonization algorithm, our final algorithm from the previous theorem does not lead to canonical forms. However, this is the only part that depends on their isomorphism algorithm.

8 Outlook and Open Questions

One could ask the question whether our isomorphism algorithm for graphs can be improved to a FPT-algorithm that runs in time $2^{\mathrm{polylog}(k)}n^{\mathcal{O}(1)}$ where n is the number of vertices and k is the maximum treewidth of the given graphs. There are various reasons why this might be difficult. One reason is that our approach would require a FPT-algorithm for the isomorphism problem of graphs of maximum degree d that runs in time $2^{\mathrm{polylog}(d)}n^{\mathcal{O}(1)}$. However, it is an open question whether any FPT-algorithm for the graph isomorphism problem parameterized by maximum degree exists. Another reason is that an algorithm for graphs running in time $2^{\mathrm{polylog}(k)}n^{\mathcal{O}(1)}$ would imply an isomorphism algorithm for hypergraphs (V, H) running in time $2^{\mathrm{polylog}|V|}|H|^{\mathcal{O}(1)}$. It is also an open question, whether such a hypergraph isomorphism algorithm exists [2]. If this were indeed the case, one could hope for an improvement of our canonization algorithm for a set J consisting of labeling cosets that runs in time $2^{\mathrm{polylog}|V|}|J|^{\mathcal{O}(1)}$.

Recently, Babai extended his quasipolynomial-time algorithm to the canonization problem for graphs [3]. With Babai's result, it is a natural question whether the bounded-degree isomorphism algorithm of [13] extends to canonization as well. The present isomorphism algorithm for graphs parameterized by treewidth should then be amenable to canonization as well.

Another question that arises is about permutation groups $G \leq \mathrm{Sym}(V)$. The canonical labeling problem for permutation groups is of great interest because it also solves the normalizer problem. In our recent work, we gave a canonization algorithm for explicitly given permutation groups running in time $2^{\mathcal{O}(|V|)}|G|^{\mathcal{O}(1)}$ [31]. Recently, the framework was extended to permutation groups that are implicitly given and the running time was improved to $2^{\mathcal{O}(|V|)}$ [35]. The present work implies a canonization algorithm running in time $(|V| + |G|)^{\mathrm{polylog}|V|}$. An important question is whether the present techniques can be combined with the canonization techniques for implicitly given permutation groups to obtain a canonization algorithm running in time $2^{\mathrm{polylog}|V|}$.

Finally, we ask whether the isomorphism problem can be solved in time $n^{\mathrm{polylog}(|V(H)|)}$ where n is the number of vertices and H is an excluded topological subgraph H of the given graphs. Even for excluded minors H, we do not have such an algorithm.

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. Groups, graphs, algorithms: The graph isomorphism problem. In *Proc. ICM*, pages 3303–3320, 2018.

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

4 László Babai and Eugene M. Luks. Canonical labeling of graphs. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 171–183, 1983. doi:10.1145/800061.808746.

5 Hans L. Bodlaender. Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. *J. Algorithms*, 11(4):631–643, 1990. doi:10.1016/0196-6774(90)90013-5.

6 Hans L. Bodlaender, Leizhen Cai, Jianer Chen, Michael R. Fellows, Jan Arne Telle, and Dániel Marx. Open problems in parameterized and exact computation-iwpec 2006. *UU-CS*, 2006, 2006.

7 Adam Bouland, Anuj Dawar, and Eryk Kopczynski. On tractable parameterizations of graph isomorphism. In *Parameterized and Exact Computation - 7th International Symposium, IPEC 2012, Ljubljana, Slovenia, September 12-14, 2012. Proceedings*, pages 218–230, 2012. doi:10.1007/978-3-642-33293-7_21.

8 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

9 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

10 I. S. Filotti and Jack N. Mayer. A polynomial-time algorithm for determining the isomorphism of graphs of fixed genus (working paper). In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*, pages 236–243, 1980. doi:10.1145/800141.804671.

11 Martin Grohe. Isomorphism testing for embeddable graphs through definability. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 63–72, 2000. doi:10.1145/335305.335313.

12 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.

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

14 Martin Grohe, Daniel Neuen, Pascal Schweitzer, and Daniel Wiebking. An improved isomorphism test for bounded-tree-width graphs. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 67:1–67:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.67.

15 Harald Andrés Helfgott. Isomorphismes de graphes en temps quasi-polynomial (d'après babai et luks, weisfeiler-leman...), 2017. arXiv:1701.04372.

16 John E. Hopcroft and Robert Endre Tarjan. A v^2 algorithm for determining isomorphism of planar graphs. *Inf. Process. Lett.*, 1(1):32–34, 1971. doi:10.1016/0020-0190(71)90019-6.

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

18 Ken-ichi Kawarabayashi and Bojan Mohar. Graph and map isomorphism and all polyhedral embeddings in linear time. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 471–480, 2008. doi:10.1145/1374376.1374443.

19 Stefan Kratsch and Pascal Schweitzer. Isomorphism for graphs of bounded feedback vertex set number. In *Algorithm Theory - SWAT 2010, 12th Scandinavian Symposium and Workshops on Algorithm Theory, Bergen, Norway, June 21-23, 2010. Proceedings*, pages 81–92, 2010. doi:10.1007/978-3-642-13731-0_9.

20 Hanns-Georg Leimer. Optimal decomposition by clique separators. *Discrete Mathematics*, 113(1-3):99–123, 1993. doi:10.1016/0012-365X(93)90510-Z.

21 Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Fixed-parameter tractable canonization and isomorphism test for graphs of bounded treewidth. *SIAM J. Comput.*, 46(1):161–189, 2017. doi:10.1137/140999980.

22 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.

23 Gary L. Miller. Graph isomorphism, general remarks. *J. Comput. Syst. Sci.*, 18(2):128–142, 1979. doi:10.1016/0022-0000(79)90043-6.

24 Gary L. Miller. Isomorphism testing for graphs of bounded genus. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*, pages 225–235, 1980. doi:10.1145/800141.804670.

25 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.

26 Wendy Myrvold and William L. Kocay. Errors in graph embedding algorithms. *J. Comput. Syst. Sci.*, 77(2):430–438, 2011. doi:10.1016/j.jcss.2010.06.002.

27 Daniel Neuen. Hypergraph isomorphism for groups with restricted composition factors. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (virtual conference)*, 2020. To appear.

28 Yota Otachi. Isomorphism for graphs of bounded connected-path-distance-width. In *Algorithms and Computation - 23rd International Symposium, ISAAC 2012, Taipei, Taiwan, December 19-21, 2012. Proceedings*, pages 455–464, 2012. doi:10.1007/978-3-642-35261-4_48.

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

30 Neil Robertson and Paul D. Seymour. Graph minors. i. excluding a forest. *J. Comb. Theory, Ser. B*, 35(1):39–61, 1983. doi:10.1016/0095-8956(83)90079-5.

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

32 Uwe Schöning. Graph isomorphism is in the low hierarchy. *J. Comput. Syst. Sci.*, 37(3):312–323, 1988. doi:10.1016/0022-0000(88)90010-4.

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

34 Louis Weinberg. A simple and efficient algorithm for determining isomorphism of planar triply connected graphs. *IEEE Transactions on Circuit Theory*, 13(2):142–148, 1966.

35 Daniel Wiebking. Normalizes and permutational isomorphisms in simply-exponential time. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 230–238, 2020. doi:10.1137/1.9781611975994.14.

36 Koichi Yamazaki, Hans L. Bodlaender, Babette de Fluiter, and Dimitrios M. Thilikos. Isomorphism for graphs of bounded distance width. *Algorithmica*, 24(2):105–127, 1999. doi:10.1007/PL00009273.

37 Viktor N. Zemlyachenko. Canonical numbering of trees. In *Proc. Seminar on Comb. Anal. at Moscow State University*, page 55, 1970.

38 Viktor N. Zemlyachenko, Nickolay M. Korneenko, and Regina I. Tyshkevich. Graph isomorphism problem. *Journal of Soviet Mathematics*, 29(4):1426–1481, 1985.