Computer Science Review 60 (2026) 100918

Contents lists available at ScienceDirect

Computer Science Review

journal homepage: www.elsevier.com/locate/cosrev

Parameterized complexity of graph isomorphism testing

Daniel Neuen[1]

University of Regensburg, Regensburg, Germany

ARTICLE INFO

Keywords:
Graph isomorphism
Parameterized complexity

ABSTRACT

We survey recent developments in the parameterized complexity of the graph isomorphism problem. Our focus lies on the fixed-parameter tractability of isomorphism testing for various graph parameters. Additionally, following Babai's quasipolynomial time isomorphism test, there has been a series of results obtaining isomorphism algorithms running in time $n^{\text{polylog}(k)}$ for different graph parameters k.

We highlight the main technical ideas underlying these algorithms and state various open questions for future research.

1. Introduction

Determining the computational complexity of the Graph Isomorphism Problem (GI) is regarded as a major open problem in theoretical computer science. It was already stated as an open problem in Karp's seminal paper on the NP-completeness of combinatorial problems in 1972 [1], and has remained one of the few natural problems in NP that are neither known to be NP-complete nor known to be solvable in polynomial time. In 2016, Babai [2] proved in a breakthrough result that GI can be solved in quasipolynomial time $n^{\text{polylog}(n)} := n^{(\log n)^{\mathcal{O}(1)}}$, where n denotes the number of vertices of the input graphs. This was the first improvement on the worst-case running time of a general graph isomorphism algorithm over the $2^{\mathcal{O}\left(\sqrt{n \log n}\right)}$ [3] time bound established more than three decades earlier.

Research on GI as a computational problem can be traced back to work on chemical information systems in the 1950's (see, e.g., [4]). The first papers in the computer science literature, proposing various heuristic approaches, appeared in the 1960's (e.g., [5]). An early breakthrough was Hopcroft and Tarjan's $O(n \log n)$ isomorphism test for planar graphs [6]. Isomorphism algorithms for specific graph classes have been an active research strand ever since (e.g., [7–15]) which, in particular, resulted in an extensive study of the parameterized complexity of GI for various different graph parameters.

An important early result in this context is Luks algorithm [7] which tests isomorphism of graphs of maximum degree d in time $n^{O(d)}$. This work introduced a divide-and-conquer approach along the structure of the permutation groups involved which sparked research on GI in the 1980's [3,16] and it also forms the foundation of Babai's quasipolynomial time isomorphism test [2]. In the following decades, polynomial-time isomorphism tests were developed for a variety of different graph classes, including graphs of bounded eigenvalue multiplicity [8], bounded genus [9,10], bounded tree-width [12], bounded rank-width [15], and classes that exclude a fixed graph as a minor [13] or even only as a topological subgraph [14].

However, all these algorithms have in common that the parameter in question appears in the exponent of the running time, i.e., in the language of parameterized complexity, they are *XP algorithms*. In the area of parameterized complexity, developed by Downey and Fellows in the 1990's [17,18], we are generally aiming for *fpt algorithms* running in time $f(k) \cdot n^c$ where k is the parameter in question, f is a computable function and c is a fixed constant (not depending on k). In fact, already the first book by Downey and Fellows [17] asks for fpt algorithms for GI parameterized by the maximum degree and the tree-width.

One of the earliest positive results is an fpt isomorphism test parameterized by the eigenvalue multiplicity k running in time $2^{O(k \log k)} n^{O(1)}$ by Evdokimov and Ponomarenko [19].[1] Other early examples include fpt algorithms parameterized by the feedback vertex number [21] or the tree-depth [22]. Still, for many natural parameterizations of GI, fpt algorithms remain elusive to this day (e.g., for maximum degree) or have only been obtained in recent years. In 2014, Lokshtanov, Pilipczuk, Pilipczuk and Saurabh obtained the first fpt isomorphism test parameterized by tree-width k running in time $2^{O(k^5 \log k)} n^5$ [23] which was later improved to a running time of $2^{k \,\text{polylog}(k)} n^3$ [24]. Additionally,

Email address: daniel.neuen@tu-dresden.de.

[1] Present address: TU Dresden, Dresden, Germany.

[2] In [19], the running time $2^{O(k \log k)} n^{O(1)}$ is achieved under the Classification of Finite Simple Groups. An alternative algorithm with the same running time was obtained in [20].

https://doi.org/10.1016/j.cosrev.2026.100918
Received 7 May 2024; Received in revised form 8 December 2025; Accepted 20 January 2026
Available online 5 February 2026

an fpt algorithm for GI parameterized by the genus g running in time $2^{O(g^4 \log g)}n^{O(1)}$ was obtained in [25]. Recently, Lokshtanov, Pilipczuk, Pilipczuk and Saurabh generalized both of these results to obtain an fpt isomorphism test[2] parameterized by the size of a forbidden minor [26].

Following Babai's breakthrough result [2], another strand of work aimed at obtaining parameterized quasipolynomial isomorphism tests running in time $n^{\text{polylog}(k)}$ for certain graph parameters k. In a joint work with Martin Grohe and Pascal Schweitzer, we obtained an isomorphism test running in time $n^{\text{polylog}(d)}$ for graphs of maximum degree d which extends both Babai's quasipolynomial time algorithm [2] and Luks isomorphism test for graphs of bounded degree [7]. In a series of follow-up works, this result was extended to graphs of bounded genus g [27], bounded tree-width k [28], and excluding a minor [29] or even only a topological subgraph [30] on h vertices. Note that a running time of the form $n^{\text{polylog}(k)}$ is incomparable to an fpt algorithm with a running time of the form $2^{O(k^c)}n^{O(1)}$ for some constant c (or similar) which is common for most of the available fpt isomorphism tests.

A problem closely related to GI is the Graph Canonization Problem (Can). A *canonization* for a class of graphs C is a mapping $\kappa\colon C \to C$ such that $\kappa(G) \cong G$ for all $G \in C$, and $\kappa(G) = \kappa(H)$ for all isomorphic $G, H \in C$. Intuitively speaking, a canonization gives a "standard representation" of a graph $G \in C$ so that two graphs are isomorphic if and only if their standard representations are *equal*. The Graph Canonization Problem (Can) asks, given a graph G, to compute $\kappa(G)$ for some fixed canonization κ. Clearly, GI reduces to Can, but a reduction in the other direction is not known. Still, many algorithms for GI can be modified to compute a graph canonization within the same running time, although this often comes at the price of additional technical complications. For example, by appropriately extending the tools underlying his algorithm, Babai obtained a quasipolynomial time canonization algorithm [31]. We stress that the same is possible for many of the algorithms discussed in this survey. However, to keep the presentation simple, we only focus on isomorphism testing in the remainder of this work.

Finally, we would also like to stress that this survey focuses on the parameterized complexity of isomorphism testing. As a result, there are many research strands on GI that we (almost) completely ignore (e.g., practical isomorphism algorithms). We refer the reader to [32] for a broader survey. Another survey on recent developments on GI can also be found in [33].

2. Preliminaries

2.1. Graphs

Let us review some basic notation. We use $\mathbb{N}$ to denote the natural numbers and write $[k]$ for the set $\{1, \dots, k\}$. We use standard graph notation. An (undirected) graph is a pair $G = (V, E)$ with a finite vertex set V and an edge set $E \subseteq \{\{v, w\} \mid v, w \in V\}$. We write $V(G)$ and $E(G)$ to denote the vertex and edge set of a graph G, respectively. We use vw as shorthand to denote the set $\{v, w\}$. The *(open) neighborhood* of a vertex $v \in V(G)$ is the set $N_G(v) := \{w \in V(G) \mid vw \in E(G)\}$. For a set $X \subseteq V(G)$ we write $N_G(X) = (\bigcup_{v \in X} N_G(v)) \setminus X$ for the neighborhood of X. The *degree* of v is $\deg_G(v) := |N_G(v)|$. The *maximum degree* of a graph G is $\Delta(G) := \max_{v \in V(G)} \deg_G(v)$.

Let G be a graph and $A, B \subseteq V(G)$. We write $E_G(A, B) := \{vw \in E(G) \mid v \in A, w \in B\}$ to denote the set of edges with one endpoint in A, and one endpoint in B. The *induced subgraph* of G with vertex set A is the graph $G[A] := (A, E_G(A, A))$. We also write $G - A := G[V(G) \setminus A]$ to denote the subgraph induced by the complement of A.

Two graphs G_1 and G_2 are *isomorphic*, denoted by $G_1 \cong G_2$, if there is a bijection $\varphi\colon V(G_1) \to V(G_2)$ such that $vw \in E(G_1)$ if and only if $\varphi(v)\varphi(w) \in E(G_2)$ for all $v, w \in V(G_1)$. In this case, the mapping φ is an *isomorphism* from G_1 to G_2. This definition gives rise to the following computational problem which is the main topic of this article.

Graph Isomorphism Problem (GI)
Input: Two graphs G_1, G_2.
Question: Decide whether $G_1 \cong G_2$.

In the context of isomorphism testing, it is often convenient to work with *(vertex-)colored* graphs $G = (V, E, \lambda)$ where (V, E) is a graph and $\lambda\colon V(G) \to C$ is a *coloring* of the vertices where C is some finite set of colors. For a vertex $v \in V(G)$, we write $[v]_\lambda := \{w \in V(G) \mid \lambda(v) = \lambda(w)\}$ to denote the *color class* that v is contained in. Isomorphisms between colored graphs must respect the colors of the vertices.

2.2. Group theory

Next, we provide some basic notation on permutation groups. For a general background on group theory we refer to [34], whereas background on permutation groups can be found in [35]. Additionally, basic facts on algorithms for permutation groups are given in [36].

A *permutation group* acting on a set Ω is a subgroup $\Gamma \leq \text{Sym}(\Omega)$ of the symmetric group. The size of the permutation domain Ω is called the *degree* of Γ. If $\Omega = [n]$, then we also write S_n instead of $\text{Sym}(\Omega)$. For an element $\alpha \in \Omega$ and a permutation $\gamma \in \text{Sym}(\Omega)$ we write α^γ to denote the image $\gamma(\alpha)$ of α under the permutation γ. We apply permutations from left to right, i.e., $\alpha^{(\gamma\delta)} = (\alpha^\gamma)^\delta$ for permutations $\gamma, \delta \in \text{Sym}(\Omega)$.

A set $S \subseteq \Gamma$ is a *generating set* for Γ if for every $\gamma \in \Gamma$ there exist $\delta_1, \dots, \delta_k \in S$ such that $\gamma = \delta_1 \dots \delta_k$. For $S \subseteq \text{Sym}(\Omega)$ we write $\langle S \rangle$ for the permutation group generated by S. In order to perform computational tasks for permutation groups efficiently the groups are represented by generating sets of small size (i.e., polynomial in the size of the permutation domain). Indeed, most algorithms are based on so-called *strong generating sets*, which can be chosen to be of size quadratic in the size of the permutation domain of the group and can be computed in polynomial time given an arbitrary generating set (see, e.g., [36]). In particular, every permutation group has a generating set of quadratic size.

2.3. Parameterized complexity

We assume the reader is familiar with basic concepts in complexity theory. We say a computational problem Π is *GI-complete* if it is equivalent to the Graph Isomorphism Problem under polynomial-time many-one reductions.

We give a very brief introduction to parameterized complexity. For more background we refer the reader to [17,18,37,38]. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$ where Σ is a fixed finite alphabet. For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, we refer to k as the *parameter* of the instance. A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is *fixed-parameter tractable (fpt)* if there is an algorithm $\mathcal{A}$, a computable function f and a constant c such that, given an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, $\mathcal{A}$ decides whether $(x, k) \in L$ in time $f(k) \cdot (|x| + k)^c$.

Observe that the function f in the definition needs to be computable. We sometimes say a problem (or an algorithm for the problem) is *strongly uniform fpt* to highlight the fact that f is a computable function. This is motivated by the following two variants. We say that a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is

- *weakly uniform fpt* if there is an algorithm $\mathcal{A}$, a (not necessarily computable) function f and a constant c such that, given an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, $\mathcal{A}$ decides whether $(x, k) \in L$ in time $f(k) \cdot (|x| + k)^c$.
- *non-uniform fpt* if there is a function f and a constant c such that for every k there is an algorithm $\mathcal{A}_k$ that, given an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, decides whether $(x, k) \in L$ in time $f(k) \cdot (|x| + k)^c$.

[3] We remark that their isomorphism test is strictly speaking not an fpt algorithm; see Section 5.2 for details.

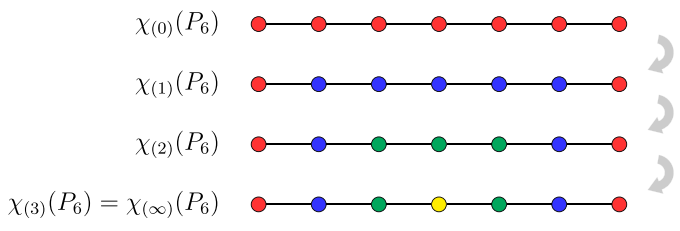

Fig. 1. The iterations of the Color Refinement algorithm on a path of length 6. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

We stress that most of the algorithms we encounter in this article are strongly uniform, but there are cases where we only obtain a weakly uniform or non-uniform fpt algorithms.

Let $L_1 \subseteq \Sigma_1^* \times \mathbb{N}$ and $L_2 \subseteq \Sigma_2^* \times \mathbb{N}$ denote parameterized problems. An *fpt reduction* from L_1 to L_2 is an algorithm that maps each instance $(x_1, k_1) \in \Sigma_1^* \times \mathbb{N}$ to an instance $(x_2, k_2) \in \Sigma_1^* \times \mathbb{N}$ such that $(x_1, k_1) \in L_1$ if and only if $(x_2, k_2) \in L_2$, $k_2 \leq g(k_1)$ for some computable function g, and $\mathcal{A}$ runs in time $f(k_1) \cdot (|x_1| + k_1)^c$ for some computable function f and a constant c.

Finally, let us note that throughout this work, we are not only interested in fpt algorithms, but also *XP algorithms* where the running time is of the form $f(k) \cdot n^{g(k)}$ for computable functions f, g. In particular, we are interested in algorithms with a running time of the form $n^{\text{polylog}(k)}$ where $\text{polylog}(k)$ is some polynomial function in $\log k$.

3. The Weisfeiler-Leman algorithm

In this section, we discuss the Weisfeiler-Leman (WL) algorithm which forms a key heuristic for testing isomorphism of graphs and is used as a subroutine in many algorithms.

3.1. The color refinement algorithm

One of the most basic procedures to tackle the Graph Isomorphism Problem is the *Color Refinement algorithm* which computes a coloring of the vertices based on iterated degree sequences. More precisely, an initially uniform coloring is iteratively refined by counting, for each color, the number of neighbors of that color.

Let $\lambda_1, \lambda_2 \colon V(G) \to C$ be two colorings of the vertices of a graph G. We say that λ_1 refines λ_2, denoted by $\lambda_1 \preceq \lambda_2$, if $\lambda_2(v) = \lambda_2(w)$ for all $v, w \in V(G)$ for which $\lambda_1(v) = \lambda_1(w)$. The two colorings are *equivalent*, denoted $\lambda_1 \equiv \lambda_2$, if $\lambda_1 \preceq \lambda_2$ and $\lambda_2 \preceq \lambda_1$.

The Color Refinement algorithm takes as input a vertex-colored graph $G = (V, E, \lambda)$ and computes an isomorphism-invariant coloring of the vertices. Initially, the algorithm sets $\chi_{(0)}(G) := \lambda$. The initial coloring is iteratively refined as follows. For $i > 0$ we set $\chi_{(i)}(G)(v) := \big(\chi_{(i-1)}(G)(v); \mathcal{M}_i(v)\big)$ where

$$\mathcal{M}_i(v) := \left\{\!\!\left\{ \chi_{(i-1)}(G)(w) \;\middle|\; w \in N_G(v) \right\}\!\!\right\}$$

is the multiset of colors that appear in the neighborhood of v. By definition, we have that $\chi_{(i)}(G) \preceq \chi_{(i-1)}(G)$ for every $i > 0$. Hence, there is some minimal $i_\infty \geq 0$ such that $\chi_{(i_\infty)}(G) \equiv \chi_{(i_\infty+1)}(G)$. We call $\chi_{(i_\infty)}(G)$ the *stable* coloring and denote it by $\chi_{(\infty)}(G)$. As an example, the sequence of colorings computed for the path of length 6 is given in Fig. 1.

Given a graph G the stable coloring $\chi_{(\infty)}(G)$ can be computed[3] in almost linear time $O((n+m)\log n)$ [39]. Due to its efficiency, the Color Refinement algorithm is used in essentially all practical tools for isomorphism testing [40–44]. On the other hand, it is also limited in its power since, e.g., on regular graphs it always returns the uniform coloring (assuming the vertex-coloring λ of G is uniform).

3.2. The Weisfeiler-Leman algorithm

For $k \geq 2$ the *k-dimensional Weisfeiler-Leman algorithm (k-WL)* is a natural generalization of the Color Refinement algorithm which, instead of coloring vertices, computes a coloring of k-tuples of vertices. The 2-dimensional version was introduced by Weisfeiler and Leman [45]. The generalization to higher dimensions was independently introduced by Babai and Mathon as well as Immerman and Lander [46] (see [2] for a historical note).

Let us fix some $k \geq 2$. To introduce k-WL, we need the notion of the *atomic type* $\text{atp}(G, \bar{v})$ of a tuple $\bar{v} = (v_1, \dots, v_k)$ of vertices in a graph G. Intuitively speaking, it describes the isomorphism type of the labeled induced subgraph $G[\{v_1, \dots, v_k\}]$. More formally, for graphs G, H and tuples $\bar{v} = (v_1, \dots, v_k) \in (V(G))^k, \bar{w} = (w_1, \dots, w_k) \in (V(H))^k$ we have $\text{atp}(G, \bar{v}) = \text{atp}(H, \bar{w})$ if and only if the mapping $v_i \mapsto w_i$ is an isomorphism from $G[v_1, \dots, v_k]$ to $H[w_1, \dots, w_k]$. Technically, one can view $\text{atp}(G, v_1, \dots, v_k)$ as a pair of Boolean $(k \times k)$-matrices, one describing equalities between the v_i, and one describing adjacencies. If G is a colored graph, we add a diagonal matrix for the vertex coloring.

Now, k-WL computes a sequence of colorings $\chi^k_{(i)}(G)$ of $(V(G))^k$ as follows. Initially, each tuple is colored by its atomic type, i.e., we set $\chi^k_{(0)}(G)(\bar{v}) := \text{atp}(G, \bar{v})$. For $i > 0$, we set $\chi^k_{(i)}(G)(\bar{v}) := \big(\chi^k_{(i-1)}(G)(\bar{v}); \mathcal{M}_i(\bar{v})\big)$ where

$$\mathcal{M}_i(\bar{v}) := \left\{\!\!\left\{ \big(\chi^k_{(i-1)}(G)(\bar{v}[w/1]), \dots, \chi^k_{(i-1)}(G)(\bar{v}[w/k])\big) \;\middle|\; w \in V(G) \right\}\!\!\right\}$$

and $\bar{v}[w/i]$ denotes the k-tuple $(v_1, \dots, v_{i-1}, w, v_{i+1}, \dots, v_k)$ obtained from $\bar{v}$ by replacing the i-th entry with w. As before, there is a minimal i_∞ such that $\chi^k_{(i_\infty)}(G) \equiv \chi^k_{(i_\infty+1)}(G)$, and we denote $\chi^k_{(\infty)}(G) := \chi^k_{(i_\infty)}(G)$ as the *k-stable coloring*. As an example, the sequence of colorings computed by 2-WL for the cycle of length 9 is given in Fig. 2. For every $k \geq 2$ the k-stable coloring of an n-vertex graph can be computed in time $O(n^{k+1} \log n)$ [46].

3.3. The WL dimension

To use the WL algorithm as an isomorphism test, one can compare the color patterns computed on two input graphs G_1 and G_2. If they differ, the graphs cannot be isomorphic. We say that k-WL *distinguishes* two graphs G and H if there is a color c such that

$$\left|\left\{ \bar{v} \in (V(G))^k \;\middle|\; \chi^k_{(\infty)}(G)(\bar{v}) = c \right\}\right| \neq \left|\left\{ \bar{w} \in (V(H))^k \;\middle|\; \chi^k_{(\infty)}(H)(\bar{w}) = c \right\}\right|$$

Observe that k-WL is not a complete isomorphism test, i.e., there are non-isomorphic graphs which are not distinguished by k-WL. For example, any pair of non-isomorphic strongly regular graphs with the same parameters is not distinguished by 2-WL. For arbitrary $k \geq 2$, the following celebrated result provides such pairs of graphs.

Theorem 3.1 (Cai, Fürer, Immerman [47]). *For every $k \geq 2$ there exist non-isomorphic 3-regular graphs G_k, H_k with $|V(G_k)| = |V(H_k)| = O(k)$ that are not distinguished by k-WL.*

This means the WL algorithm alone is unable to solve the isomorphism problem in general unless the dimension grows linearly with the number of vertices which, however, is prohibitive both in terms of running time and memory consumption. Still, for many restricted types of graphs, the WL algorithm serves as a complete isomorphism test. We say k-WL *identifies* a graph G if it distinguishes G from every other non-isomorphic graph. The *WL dimension* of G is the least $k \geq 1$ such that k-WL identifies G. For example, it is known that all planar graphs have WL dimension at most 3 [48].

More generally, Grohe [49] proved that every graph class excluding a fixed graph as a minor has bounded WL dimension. A graph H is a *minor* of a graph G if H can be obtained from G by deleting vertices and

[4] To be precise, a coloring equivalent to $\chi_{(\infty)}(G)$ is computed since the (nested) multisets become too large.

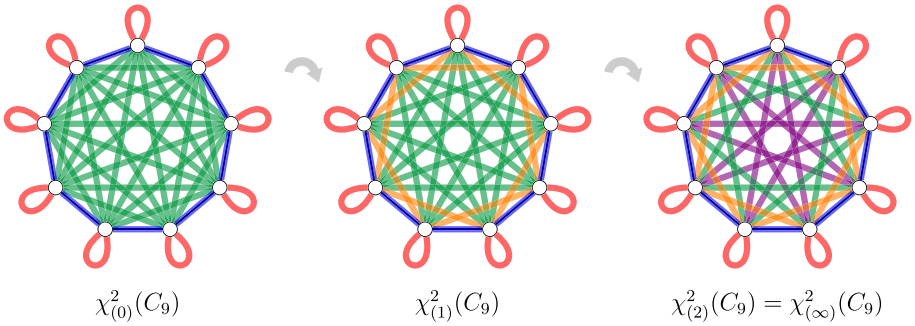

Fig. 2. The iterations of 2-WL on a cycle of length 9. In this example, $\chi^2_{(i)}(G)(v,w) = \chi^2_{(i)}(G)(w,v)$ for all $v,w \in V(G)$ and thus, all colors are represented by undirected edges. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

edges, and contracting edges (i.e., merging the two endpoints of an edge and deleting any loops or multiedges). A graph G *excludes* H *as a minor* if G has no minor isomorphic to H. For example, Wagner's theorem states that a graph is planar if and only if it excludes $K_{3,3}$ and K_5 as minors.

Theorem 3.2 (Grohe [49]). *There is a computable function f such that every graph G excluding K_h as a minor has WL dimension at most $f(h)$.*

In particular, this means that the Graph Isomorphism Problem can be solved in time $n^{O(f(h))}$ for all graphs that exclude K_h as a minor.

While this provides us with a polynomial-time isomorphism test for a wide collection of graph classes, the running time is rather unsatisfactory. In later sections, we explore how the running time can be improved by incorporating other tools such as group-theoretic and graph-theoretic methods. Indeed, while the WL dimension is bounded in terms of various graph parameters, this usually does not lead to the most efficient algorithms (in particular, no fpt complexity is achieved). Instead, we can often combine the Weisfeiler-Leman algorithm with other tools to design fast isomorphism algorithms.

Still, let us stress at this point that obtaining bounds on the WL dimension (such as in Theorem 3.2) is an important and active research area due to the connections of WL to various other areas of (theoretical) computer science such as descriptive complexity theory. We refer to [33,50,51] for a more well-rounded overview of the WL algorithm which is beyond the scope of this article.

4. The group-theoretic GI machinery

Next, we give an overview of the group-theoretic graph isomorphism machinery. This machinery dates back to Luks polynomial-time isomorphism test for graphs of bounded degree [7], and also forms the foundation of Babai's quasipolynomial-time algorithm [2] for GI. In recent years, the group-theoretic methods have also become a wide-spread tool for the design of parameterized algorithms for isomorphism testing.

In the following, we introduce the key computational problems considered within this framework and present several important algorithmic results. Afterward, we introduce the notion of t-*CR-bounded* graphs which serves as a gateway to apply group-theoretic methods for various parameterizations of GI. Fig. 3 provides an overview of the problems considered in this section and their relation to one another. We also refer to [33] for a more thorough overview of the group-theoretic techniques.

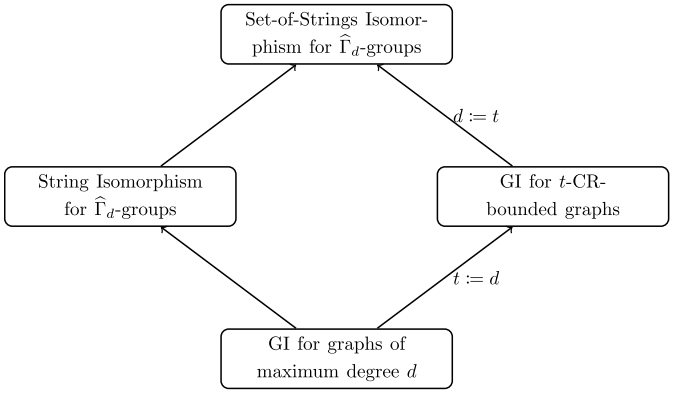

Fig. 3. Dependencies between problems tackled by group-theoretic approaches. Each solid edge represents a polynomial-time Turing reduction where the parameter is set according to the edge label.

4.1. Group-theoretic problems

In order to apply group-theoretic methods to the Graph Isomorphism Problem, Luks [7] introduced a more general problem that allows us to build recursive algorithms along the structure of the permutation groups involved. Here, we follow the notation used by Babai [2] for describing his quasipolynomial-time isomorphism test.

Let Σ be a finite set which is referred to as the *alphabet*. A *string* is a mapping $\mathfrak{x} : \Omega \to \Sigma$ from a finite ground set Ω into the alphabet Σ. Let $\gamma \in \mathrm{Sym}(\Omega)$ be a permutation of the ground set Ω. The permutation can be applied to the string $\mathfrak{x}$ to obtain $\mathfrak{x}^\gamma$ defined via

$$\mathfrak{x}^\gamma(\alpha) = \mathfrak{x}(\alpha^{\gamma^{-1}}).$$

Let $\mathfrak{y} : \Omega \to \Sigma$ be another string. The permutation γ is an *isomorphism* from $\mathfrak{x}$ to $\mathfrak{y}$ if $\mathfrak{x}^\gamma = \mathfrak{y}$. We write $\gamma : \mathfrak{x} \cong \mathfrak{y}$ to denote that γ is an isomorphism from $\mathfrak{x}$ to $\mathfrak{y}$. Given a permutation group $\Gamma \leq \mathrm{Sym}(\Omega)$, a Γ-*isomorphism* from $\mathfrak{x}$ to $\mathfrak{y}$ is a permutation $\gamma \in \Gamma$ such that $\gamma : \mathfrak{x} \cong \mathfrak{y}$. We write $\mathfrak{x} \cong_\Gamma \mathfrak{y}$ to denote that there is a Γ-isomorphism from $\mathfrak{x}$ to $\mathfrak{y}$. These definitions lead to the following computational problem.

String Isomorphism Problem (SI)
Input: A set $S \subseteq \mathrm{Sym}(\Omega)$, and two strings $\mathfrak{x}_1, \mathfrak{x}_2 : \Omega \to \Sigma$.
Question: Decide whether $\mathfrak{x}_1 \cong_\Gamma \mathfrak{x}_2$ where $\Gamma = \langle S \rangle$.

Note that the group Γ is given via a generating set which may be significantly smaller than the generated group. Indeed, the reader is encouraged to think of $|S|$ as being polynomially bounded in $|\Omega|$, whereas $|\Gamma|$ is exponential in $|\Omega|$.

Theorem 4.1. *There is a polynomial-time many-one reduction from the Graph Isomorphism Problem to the String Isomorphism Problem.*

Proof. Let G_1, G_2 denote the input graphs for GI. By renaming vertices, we may assume without loss of generality that $V := V(G_1) = V(G_2)$. We set $\Omega := \{vw \mid v \neq w \in V\}$ and $\Sigma := \{0,1\}$. Also, we define $\mathfrak{x}_1(vw) = 1$ if $vw \in E(G_1)$, and $\mathfrak{x}_1(vw) = 0$ otherwise (see Fig. 4). The string $\mathfrak{x}_2$ is defined analogously for the graph G_2. Finally, we define Γ to be the

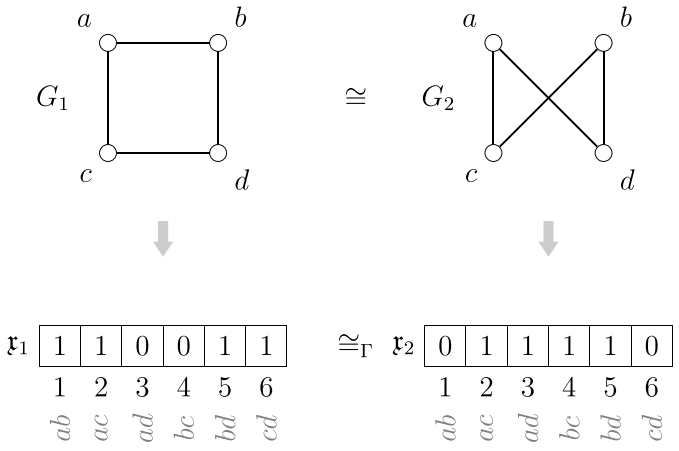

Fig. 4. Reduction from the Graph Isomorphism Problem to the String Isomorphism Problem. The two input graphs G_1 and G_2 have vertex set $V = \{a, b, c, d\}$. We set $\Omega = \{1, \dots, 6\}$ and each element corresponds to a two-element subset of V. Also we set $S = \{\gamma = (2\,4)(3\,5), \delta = (1\,5\,6\,2)(3\,4)\}$. For example, $\delta^{-1}\gamma\delta = (1\,3)(4\,6)$ is a Γ-isomorphism from $\mathfrak{x}_1$ to $\mathfrak{x}_2$ where $\Gamma = \langle S \rangle$.

permutation group obtained from the natural action of $\mathrm{Sym}(V)$ on the set of two-element subsets Ω. Then $G_1 \cong G_2$ if and only if $\mathfrak{x}_1 \cong_\Gamma \mathfrak{x}_2$. Also, the group Γ can be generated by a generating set S of size two that can be easily computed. □

Theorem 4.2 (Babai [2]). *The String Isomorphism Problem can be solved in time $n^{\mathrm{polylog}(n)}$.*

Together with Theorem 4.1 this gives a quasipolynomial-time isomorphism test for graphs.

The String Isomorphism Problem is not only useful for obtaining a general algorithm for isomorphism testing, but also in a parameterized setting. To be able to use SI in this setting, we require a suitable parameterization of (permutation) groups.

Let Γ be a group. A *subnormal series* is a sequence of normal subgroups $\Gamma = \Gamma_0 \trianglerighteq \Gamma_1 \trianglerighteq \cdots \trianglerighteq \Gamma_k = \{\mathrm{id}\}$. 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 (considered as a multiset) of factor groups (cf. [34]). A *composition factor* of a finite group Γ is a factor group of a composition series of Γ.

Definition 4.3. For $d \geq 1$ let $\hat{\Gamma}_d$ denote the class of all groups Γ for which every composition factor of Γ is isomorphic to a subgroup of S_d. The *composition width* of a group Γ, denoted by $\mathrm{cw}(\Gamma)$, is the minimal $d \geq 1$ such that $\Gamma \in \hat{\Gamma}_d$.

This definition may be hard to grasp for a reader without a strong background in group theory; in this case the reader is encouraged to think of Γ as being "composed" of subgroups of S_d (the precise details are not relevant in the remainder of this article).

Remark 4.4. We note that several relaxed variants of Definition 4.3 exist in the literature. For example, Babai et al. [16] only require *non-abelien* composition factors to be isomorphic to a subgroup of S_d. Our definition follows the original one by Luks [7]; see also [27,52]. Let us stress that some of the results presented below have only been proved for the more restricted definition given above.

When reducing from the Graph Isomorphism Problem to the String Isomorphism Problem, the composition width of the input group turns out to be a very useful parameter that naturally appears for various different parameterizations of graphs (see Section 5). This connection was first observed by Luks [7] for the graph parameter *maximum degree*.

Lemma 4.5. *There is a polynomial-time Turing reduction from the Graph Isomorphism Problem for graphs of maximum degree d to the String Isomorphism Problem for groups of composition width at most $d - 1$.*

We remark that the original reduction by Luks [7] only runs in polynomial time for fixed d; the reduction stated here is from [16] (see also [53, Theorem 5.3.4]).

Also, Luks [7] proved that the String Isomorphism Problem for groups of composition width at most d can be solved in time $n^{O(d)}$ using a recursive divide-and-conquer approach on the input permutation group. Together, this provides an isomorphism test for graphs of maximum degree d running in time $n^{O(d)}$.

With the new tools developed in Babai's algorithm [2] solving the String Isomorphism Problem in quasipolynomial time, it is natural to ask whether one can also obtain improved running times parameterized by the composition width of the input group. The following result provides a positive answer and strengthens both Luks algorithm [7] and Babai's quasipolynomial-time isomorphism test.

Theorem 4.6 (Grohe, Neuen, Schweitzer [52]). *The String Isomorphism Problem for groups of composition width at most d can be solved in time $n^{\mathrm{polylog}(d)}$.*

Corollary 4.7. *The Graph Isomorphism Problem for graphs of maximum degree at most d can be solved in time $n^{\mathrm{polylog}(d)}$.*

Still, the long-standing open question of whether isomorphism testing parameterized by the maximum degree is fpt (e.g., posed in [17,18]) remains wide open. With this in mind, it may be interesting to also investigate the relationships between the problems displayed in Fig. 3 in more detail. As a concrete example, we formulate the following question.

Question 4.8. Is there an fpt-reduction from the String Isomorphism Problem parameterized by the composition width of the input group to the Graph Isomorphism Problem parameterized by the maximum degree of the input graphs?

Let us note that the most natural approach would be to translate the input group $\Gamma \leq \mathrm{Sym}(\Omega)$ into a graph G_Γ such that $\Omega \subseteq V(G_\Gamma)$ and the automorphism group of G restricted to Ω is equal to Γ. For this approach to work, it is crucial that we only ask for an fpt reduction since the alternating group on d points can only be represented by graphs of size exponential in d [54].

As already indicated above, parameterization by the composition width of the group is not only relevant when studying isomorphism of graphs of bounded degree, but the same holds true for various other natural graph parameters. Unfortunately, in many cases, the String Isomorphism Problem turns out to be too restrictive to obtain a reduction similar to Lemma 4.5. Instead, we require a generalization where instead of two strings, we receive two sets of strings in the input.

Let $\mathfrak{X} = \{\mathfrak{x}_1, \dots, \mathfrak{x}_m\}$ and $\mathfrak{Y} = \{\mathfrak{y}_1, \dots, \mathfrak{y}_m\}$ be two sets of strings where $\mathfrak{x}_i, \mathfrak{y}_i : \Omega \to \Sigma$ are strings with domain Ω over the alphabet Σ for every $i \in \{1, \dots, m\}$. Also let $\Gamma \leq \mathrm{Sym}(\Omega)$ be a permutation group. A Γ-*isomorphism* from $\mathfrak{X}$ to $\mathfrak{Y}$ is a permutation $\gamma \in \Gamma$ such that $\mathfrak{X}^\gamma = \mathfrak{Y}$ where $\mathfrak{X}^\gamma := \{\mathfrak{x}_1^\gamma, \dots, \mathfrak{x}_m^\gamma\}$. We write $\mathfrak{X} \cong_\Gamma \mathfrak{Y}$ if there is some Γ-isomorphism from $\mathfrak{X}$ to $\mathfrak{Y}$.

Set-of-Strings Isomorphism Problem (SSI)
Input: A set $S \subseteq \mathrm{Sym}(\Omega)$, and two sets $\mathfrak{X}_1, \mathfrak{X}_2$ of strings over Ω.
Question: Decide whether $\mathfrak{X}_1 \cong_\Gamma \mathfrak{X}_2$ where $\Gamma = \langle S \rangle$.

We remark that SSI can be reduced to SI in polynomial time, but it is not known whether there is an fpt reduction under parameterization by the composition width.

The following result generalizes Theorem 4.6 and improves upon an algorithm due to Miller [11] that solves SSI in time $(n+m)^{O(d)}$. It also

serves as an important subroutine for several of the algorithms described in Section 5.

Theorem 4.9 (Neuen [27]). *The Set-of-Strings Isomorphism Problem for groups of composition width at most d can be solved in time $(n+m)^{\text{polylog}(d)}$.*

4.2. Weisfeiler-Leman and small color classes

Having introduced the group-theoretic tools, we now discuss an intermediate problem that can serve as a gateway to applying the group-theoretic GI machinery. More precisely, we formulate a purely combinatorial graph parameter that allows us to capture many circumstances under which group-theoretic methods are applicable in the context of GI. A main advantage of such a formulation is that it allows us to use the powerful group-theoretic toolbox in a black-box manner without requiring any knowledge of the underlying methods.

Let $t \geq 1$ be an integer. We consider the notion of *t-CR-bounded graphs* originally introduced by Ponomarenko (under a different name) in [55]. Here, the letters 'CR' refer to the Color Refinement algorithm discussed in Section 3. Intuitively speaking, a (vertex-colored) graph G is *t-CR-bounded* if the vertex coloring of G can be turned into the discrete coloring (i.e., each vertex has its own color) by repeatedly

- performing the Color Refinement algorithm (expressed by the letters 'CR'), and
- taking a color class $[v]_\chi := \{w \in V(G) \mid \chi(w) = \chi(v)\}$ of some vertex $v \in V(G)$ of size $|[v]_\chi| \leq t$ and assigning each vertex from the class its own color.

An example is given in Fig. 5. The next definition formalizes this concept. Recall that, for a vertex-colored graph G, we write $\chi_{(\infty)}(G)$ to denote the coloring computed by the Color Refinement algorithm on G.

Definition 4.10. Let $t \geq 1$ and let $G = (V, E, \lambda)$ be a vertex-colored graph. We define the sequence $(\chi_i)_{i\geq 0}$ of colorings where $\chi_0 := \lambda$,

$$\chi_{2i+1} := \chi_{(\infty)}(V, E, \chi_{2i})$$

and

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

for all $i \geq 0$.

For the minimal $i_\infty \geq 0$ such that $\chi_{i_\infty} \equiv \chi_{i_\infty+1}$, we refer to χ_{i_∞} as the *t-CR-stable* coloring of G and denote it by $\chi_{t\text{-CR}}(G)$. The graph G is *t-CR-bounded* if $\chi_{t\text{-CR}}(G)$ is discrete.

The following lemma shows that isomorphism of t-CR-bounded graphs can be solved using the group-theoretic toolbox. We note that the same ideas are already used in [55] to obtain, for every fixed $t \geq 1$, a polynomial-time isomorphism test for t-CR-bounded graphs.

Lemma 4.11 ([27, Lemma 5.5]). *There is a polynomial-time Turing reduction from the Graph Isomorphism Problem for t-CR-bounded graphs to the Set-of-Strings Isomorphism Problem for groups with composition width at most t.*

Corollary 4.12. *The Graph Isomorphism Problem for t-CR-bounded graphs can be solved in time $n^{\text{polylog}(t)}$.*

5. Sparse graphs

In this section, we discuss a series of recent results on isomorphism testing of sparse graphs. The first set of results concerns isomorphism tests running in time $n^{\text{polylog}(k)}$ for various graph parameters k including the maximum degree, genus, and tree-width of the input graphs. To obtain these results, the notion of t-CR-bounded graphs introduced in the last section turns out to be a powerful tool. We also discuss several FPT algorithms for sparse graph parameters that have been developed in recent years. Here, additional ideas are often required since we can only make limited use of the group-theoretic GI machinery due to the restrictions on the running time. The dependencies between the problems considered in this section are shown in Fig. 6.

5.1. XP algorithms with polylogarithmic parameter dependence

We start by discussing isomorphism tests running in time $n^{\text{polylog}(k)}$ for various graph parameters k. The starting point for all of the algorithms is the notion of t-CR-bounded graphs and the corresponding isomorphism test from Corollary 4.12. Also, we regularly rely on a second standard tool which is the individualization of single vertices. Intuitively speaking, this allows us to break potential regularities in the input graph (for example, on a d-regular graph, the t-CR-stable coloring achieves no refinement unless $t \geq n$) and identify a "starting point" for analyzing the t-CR-stable coloring.

Let G be a graph and let $X \subseteq V(G)$ be a set of vertices. Let $\lambda^* \colon V(G) \to C$ be the vertex-coloring obtained from *individualizing* all vertices in the set X, i.e., $\lambda^*(v) := (v,1)$ for $v \in X$ and $\lambda^*(v) := (0,0)$ for $v \in V(G) \setminus X$. Let $\chi := \chi_{t\text{-CR}}(G, \lambda^*)$ denote the t-CR-stable coloring with respect to the input graph (G, λ^*). Recall that we use $[v]_\chi := \{w \in V(G) \mid \chi(v) = \chi(w)\}$ to denote the color class of v under the coloring χ. We define the *t-closure* of the set X (with respect to G) to be the set

$$\text{cl}_t^G(X) := \left\{ v \in V(G) \mid |[v]_\chi| = 1 \right\}.$$

Observe that $X \subseteq \text{cl}_t^G(X)$. For $v_1, \dots, v_\ell$ we also use $\text{cl}_t^G(v_1, \dots, v_\ell)$ as a shorthand for $\text{cl}_t^G(\{v_1, \dots, v_\ell\})$.

If $\text{cl}_t^G(X) = V(G)$ then there is an isomorphism test for G running in time $n^{|X|+\text{polylog}(t)}$. An algorithm first individualizes all vertices from X creating $n^{|X|}$ many instances of GI for t-CR-bounded graphs each of which can be solved using Corollary 4.12. This provides us with a generic and powerful method for obtaining polynomial-time isomorphism tests for various classes of graphs. To illustrate this approach, we first recover the isomorphism test for graphs of maximum degree d from Corollary 4.7.

Theorem 5.1 ([55]). *Let G be a connected graph of maximum degree d and let $v \in V(G)$. Then $\text{cl}_d^G(v) = V(G)$. In particular, there is a polynomial-time Turing reduction from the Graph Isomorphism Problem for graphs of maximum degree d to the Graph Isomorphism Problem for d-CR-bounded graphs.*

Proof. Let $\chi := \chi_{d\text{-CR}}(G, \lambda^*)$ denote the d-CR-stable coloring where λ^* is the coloring obtained from individualizing v. For $i \geq 0$ let $V_i := \{w \in$

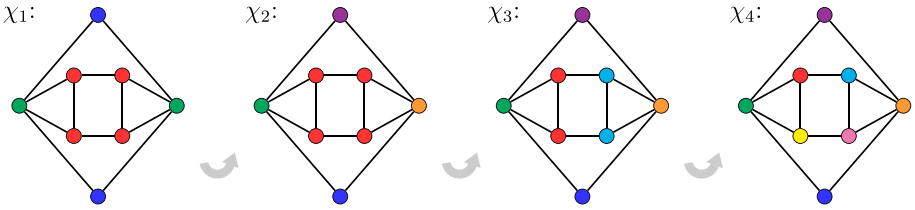

Fig. 5. Visualization of a graph G and the sequence of colorings described in Definition 4.10 for $t = 2$. The coloring χ_4 is discrete, so G is 2-CR-bounded.

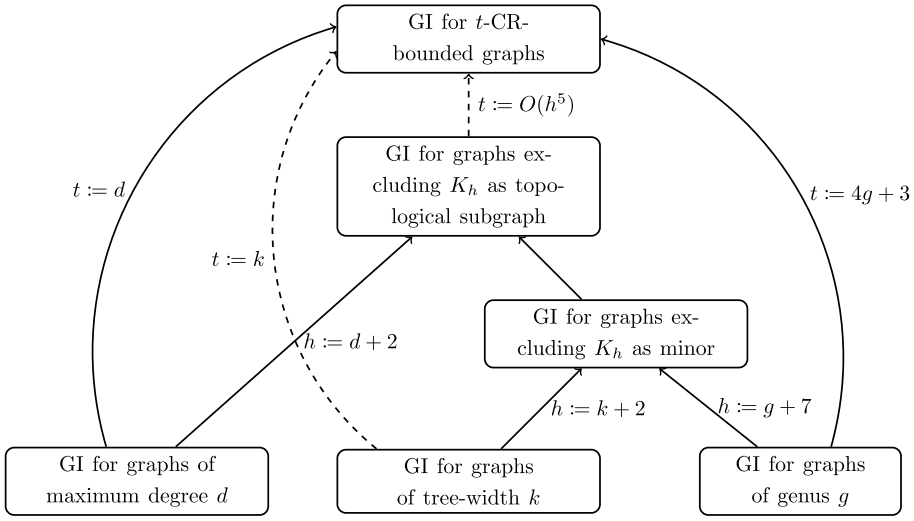

Fig. 6. Dependencies between isomorphism problems for sparse graphs. Each solid edge represents a polynomial-time Turing reduction where the parameter is set according to the edge label. The dashed edges represent fpt reductions.

$V(G) \mid \mathrm{dist}_G(v,w) \leq i\}$. We prove by induction on $i \geq 0$ that $V_i \subseteq \mathrm{cl}_d^G(v)$. Since $V_n = V(G)$ this implies that $\mathrm{cl}_d^G(v) = V(G)$.

The base case $i = 0$ is trivial since $V_0 = \{v\} \subseteq \mathrm{cl}_d^G(v)$ as already observed above. So suppose $i \geq 0$ and let $w \in V_{i+1}$. Then there is some $u \in V_i$ such that $uw \in E(G)$. Moreover, $u \in \mathrm{cl}_d^G(v)$ by the induction hypothesis. This means that $|[u]_\chi| = 1$. Since χ is stable with respect to the Color Refinement algorithm $[w]_\chi \subseteq N(u)$. So $|[w]_\chi| \leq \deg(u) \leq d$. Hence, $|[w]_\chi| = 1$ because χ is d-CR-stable. □

5.1.1. Graphs of bounded genus

Next, let us turn to a slightly more complicated example by considering graphs that can be embedded in a surface of Euler genus at most g without edge crossings. Actually, the only property we exploit is that such graphs exclude a certain minor. Recall that a graph H is a *minor* of a graph G if H can be obtained from G by deleting edges and vertices, and contracting edges. A graph G *excludes H as a minor* if G has no minor that is isomorphic to H. For our isomorphism test, we use the fact that graphs of genus g exclude $K_{3,4g+3}$ as a minor [56,57], where $K_{3,h}$ denotes the complete bipartite graph with 3 vertices on the left side and h vertices on the right side. Note that for $g = 0$ we obtain the well-known fact that planar graphs exclude $K_{3,3}$ as a minor.

Also, using well-known decompositions into 3-connected components (see, e.g., [6]), we may restrict our attention to 3-connected graphs.[4] A graph G is 3-*connected* if $G - X$ is connected for every $X \subseteq V(G)$ such that $|X| < 3$. The next lemma provides the key insight for our isomorphism test. A proof of this lemma is already implicit in [58]; an explicit version can be found in [27, Corollary 6.3].

Lemma 5.2. *Let G be a 3-connected graph that excludes $K_{3,h}$ as a minor and let $v_1, v_2, v_3 \in V(G)$ be distinct vertices. Then $\mathrm{cl}_{h-1}^G(v_1, v_2, v_3) = V(G)$.*

The lemma states that a 3-connected graph that excludes $K_{3,h}$ as a minor is $(h-1)$-CR-bounded after individualizing three vertices. In particular, it implies the following reduction.

Corollary 5.3. *There is a polynomial-time Turing reduction from the* Graph Isomorphism Problem *for graphs that exclude $K_{3,h}$ as a minor to the* Graph Isomorphism Problem *for $(h-1)$-CR-bounded graphs.*

Together with Corollary 4.12, we obtain that isomorphism of graphs that exclude $K_{3,h}$ as a minor can be tested in time $n^{\mathrm{polylog}(h)}$ and hence, isomorphism for graphs of genus g can be tested in time $n^{\mathrm{polylog}(g)}$.

Now, let us provide a proof for Lemma 5.2.

Proof (of Lemma 5.2). Let λ^* denote the coloring obtained from individualizing v_1, v_2, v_3, i.e., $\lambda^*(v_i) := (i, 1)$ for $i \in \{1,2,3\}$ and $\lambda^*(v) := (0,0)$ for $v \in V(G) \setminus \{v_1, v_2, v_3\}$. Let $\chi := \chi_{(h-1)\text{-CR}}(G, \lambda^*)$ denote the $(h-1)$-CR-stable coloring with respect to the input graph (G, λ^*).

Let $I := \mathrm{cl}_{h-1}^G(v_1, v_2, v_3)$ and suppose towards a contradiction that $I \neq V(G)$. Now let $C := \chi(V(G))$ denote the set of colors in the image of χ, and let $C_I := \chi(I)$. Consider the graph H with vertex set $V(H) := C$ and

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

Let C_Z be a connected component of $H - C_I$ and let $Z \subseteq V(G)$ be those vertices $v \in V(G)$ with $\chi(v) \in C_Z$. Observe that $N_G(Z) \subseteq I$, and $|N_G(Z)| \geq 3$ since G is 3-connected and $|I| \geq 3$. Let us fix three distinct $w_1, w_2, w_3 \in N_G(Z)$ and let $c_i := \chi(w_i)$ denote the color of w_i, $i \in \{1,2,3\}$.

Now consider a spanning tree T of $H[C_Z \cup \{c_1, c_2, c_3\}]$. Let T' be the tree obtained from T by repeatedly removing all leaves distinct from c_1, c_2, c_3. A visualization can also be found in Fig. 7. Hence, T' has exactly three leaves c_1, c_2, c_3 and a unique node c^* of degree 3. For $i \in \{1,2,3\}$ let C_i denote the set of colors on the unique path from c_i to c^* in T' including c_i, but excluding c^*. Also let $U_i := \{z \in V(G) \mid \chi(z) \in C_i\}$.

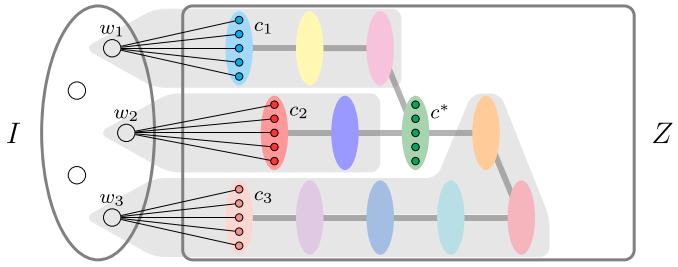

Fig. 7. The figure shows the construction of a minor $K_{3,h}$ in the proof of Lemma 5.2. The sets U_1, U_2, U_3 are marked in gray. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

[5] We remark that for this argument to work one actually needs to consider vertex- and arc-colored 3-connected graphs. For all algorithms presented in this section, one can easily incorporate the colorings. However, to keep the presentation simple, we omit the details here.

Then $G[U_i]$ is connected for every $i \in \{1,2,3\}$ and each $u \in \chi^{-1}(c^*)$ is adjacent to some vertex in U_i for all $i \in \{1,2,3\}$. Since $|\chi^{-1}(c^*)| \geq h$, we obtain a minor isomorphic to $K_{3,h}$ which is a contradiction. □

5.1.2. Graphs of bounded tree-width

After considering two simpler applications of the isomorphism test for t-CR-bounded graphs, let us now turn to the first more involved application considering graphs of small tree-width. We assume that the reader is familiar with the notion of tree-width (see, e.g., [38, Chapter 7]). Still, for the sake of completeness, let us start by providing the basic definitions. Let G be a graph. A *tree decomposition* of G is a pair (T, β) where T is a rooted tree and $\beta : V(T) \to 2^{V(G)}$, where $2^{V(G)}$ denotes the power set of $V(G)$, such that

(T.1) for every edge $vw \in E(G)$ there is some $t \in V(T)$ such that $v, w \in \beta(t)$, and
(T.2) for every $v \in V(G)$ the set $\beta^{-1}(v) := \{t \in V(T) \mid v \in \beta(t)\}$ is non-empty and connected in T, that is, $T[\beta^{-1}(v)]$ is connected.

The sets $\beta(t)$, $t \in V(T)$, are the *bags* of the decomposition. Also, for $st \in E(G)$ the sets $\beta(t) \cap \beta(s)$ are called the *adhesion sets* of the decomposition. The *width* of a tree decomposition is defined as $\max_{t \in V(T)} |\beta(t)| - 1$. The *tree-width* of a graph G, denoted $\mathrm{tw}(G)$, is the minimal width of any tree decomposition of G.

Our goal is to design efficient isomorphism tests for graphs of small tree-width. Similar to the case of graphs of bounded genus, our strategy is to decompose the graph into pieces such that each piece is t-CR-bounded after individualizing a constant number of vertices. We start by considering an illustrative example. Let $T_{d,h}$ denote a complete d-ary tree of height h, where we should think of d as being unbounded in k. Also, let $I_\ell \otimes T_{d,h}$ denote the graph obtained from $T_{d,h}$ by replacing each vertex with an independent set I^v_ℓ of size ℓ and two such sets being completely connected whenever there is an edge in $T_{d,h}$. Then $\mathrm{tw}(I_\ell \otimes T_{d,h}) \leq 2\ell$ and the graph $I_\ell \otimes T_{d,h}$ does not contain any $(\ell-1)$-separator, i.e., the graph remains connected even after removing an arbitrary set of at most $\ell - 1$ vertices. Yet, this graph is far from being t-CR-bounded even after individualizing any constant number of vertices assuming $t < d$.

To circumvent this problem we exploit *clique-separator decompositions* of *improved graphs* which already form an essential part of the first fpt isomorphism test for graphs of bounded tree-width [23].

Definition 5.4 ([23,59]). The *k-improvement* of a graph G is the graph G^k obtained from G by adding an edge between every pair of non-adjacent vertices v, w for which there are more than k pairwise internally vertex disjoint paths connecting v and w. We say that a graph G *is k-improved* when $G^k = G$.

For example, the k-improvement of $I_\ell \otimes T_{d,h}$ turns each set I^v_ℓ into a clique assuming that $k < \ell \cdot d$, since each pair of vertices $x, y \in I^v_\ell$ has $d \cdot \ell$ many common neighbors. We summarize several structural properties of G^k.

Lemma 5.5 ([23]). *Let G be a graph and $k \in \mathbb{N}$.*

1. *The k-improvement G^k is k-improved, i.e., $(G^k)^k = G^k$.*
2. *Every tree decomposition of G of width at most k is also a tree decomposition of G^k.*
3. *There exists an algorithm that, given G and k, runs in $O(k^2 n^3)$ time and either correctly concludes that $\mathrm{tw}(G) > k$, or computes G^k.*

Since the construction of G^k from G is isomorphism-invariant, the concept of the improved graph can be exploited for isomorphism testing. Given a graph, we can compute the k-improvement and assign a special color to all added edges to distinguish them from the original edges. Hence, it suffices to solve the isomorphism problem for k-improved graphs. Next, we decompose k-improved graphs along cliques.

Let G be a graph. A pair (A, B) where $A \cup B = V(G)$ is a *separation* if $E_G(A \setminus B, B \setminus A) = \emptyset$. In this case we call $A \cap B$ a *separator* and $|A \cap B|$ is the *order* of the separation. A separation (A, B) is called a *clique separation* if $A \cap B$ is a clique and $A \setminus B \neq \emptyset$ and $B \setminus A \neq \emptyset$. In this case we call $A \cap B$ a *clique separator*. We say a graph is *k-basic* if it is k-improved and does not have any separating cliques.

In order to further reduce isomorphism testing to the case of k-basic graphs, we build on a decomposition result of Leimer [60] which provides an isomorphism-invariant tree decomposition of a graph into clique-separator free parts.

Theorem 5.6 ([60,61]). *For every connected graph G there is an isomorphism-invariant tree decomposition (T, β) of G, called clique separator decomposition, such that*

1. *for every $t \in V(T)$ the graph $G[\beta(t)]$ is clique-separator free, and*
2. *each adhesion set of (T, β) is a clique.*

Moreover, given a graph G, the clique separator decomposition of G can be computed in polynomial time.

For example, applying the theorem to the k-improvement of $I_\ell \otimes T_{d,h}$, we obtain an isomorphism-invariant tree decomposition where each bag has size at most $2 \cdot \ell$, since every set I^v_ℓ is a clique separator in the k-improved graph. In general, applying the theorem to a k-improved graph results in a tree decomposition (T, β) such that $G[\beta(t)]$ is k-basic for every $t \in V(T)$. Using standard dynamic programming arguments, this decomposition allows us to essentially restrict to the case of k-basic graphs at the cost of an additional factor $2^{O(k \log k)} n^{O(1)}$ in the running time. Now, a key insight is that k-basic graphs are once again k-CR-bounded after individualizing one vertex. The next lemma provides us with the main tool for proving this.

Lemma 5.7. *Let G be a graph and let χ be a coloring that is stable with respect to the Color Refinement algorithm. Let $X_1, \dots, X_m$ be distinct color classes of χ and suppose that $E_G(X_i, X_{i+1}) \neq \emptyset$ for all $i \in [m-1]$. Then there are $\min_{i \in [m]} |X_i|$ vertex-disjoint paths from X_1 to X_m.*

To gain some intuition on the lemma, consider the simple case where $\ell := |X_i| = |X_j|$ for all $i, j \in [m]$. Since χ is stable we conclude that the bipartite graph $H_i := (X_i \cup X_{i+1}, E(X_i, X_{i+1}))$ is non-empty and biregular. Hence, by Hall's Marriage Theorem, there is a perfect matching M_i in H_i of size $|M_i| = \ell$. Taking the disjoint union of all sets M_i, $i \in [m-1]$, gives ℓ vertex-disjoint paths from X_1 to X_m. The general case can be proved using similar arguments; we refer to [29] for the details.

Lemma 5.8. *Let G be a k-basic graph. Also let $v \in V(G)$ such that $\deg(v) \leq k$. Then $\mathrm{cl}^G_k(v) = V(G)$.*

Proof. Let χ denote the k-CR-stable coloring of G after individualizing v. First note that $N(v) \subseteq \mathrm{cl}^G_k(v)$ since $\deg(v) \leq k$. Now suppose towards a contradiction that $\mathrm{cl}^G_k(v) \neq V(G)$ and let W be the vertex set of a connected component of $G - \mathrm{cl}^G_k(v)$. Then $v \notin N(W)$. Since G does not have any separating cliques there exist $w_1, w_2 \in N(W)$ such that $w_1 w_2 \notin E(G)$. Let $u_0, u_1, \dots, u_m, u_{m+1}$ be a path from $w_1 = u_0$ to $w_2 = u_{m+1}$ such that $u_i \in W$ for all $i \in [m]$. Let $X_i := [u_i]_\chi$ and note that $|X_i| \geq k+1$. Also note that $X_1 \subseteq N(w_1)$ and $X_m \subseteq N(w_2)$ since $w_1, w_2 \in \mathrm{cl}^G_k(v)$ and χ is stable. Hence, by Lemma 5.7, there are $k+1$ internally vertex-disjoint paths from w_1 to w_2. But this contradicts G being k-improved. □

Overall, this provides an isomorphism test for graphs of tree-width k running in time $2^{O(k \log k)} n^{\mathrm{polylog}(k)}$ as follows. First, we compute the clique-separator decomposition of the k-improved graph using Lemma 5.5 and Theorem 5.6. The isomorphism problem for the k-basic parts can be solved in time $n^{\mathrm{polylog}(k)}$ by Lemma 5.8 and Corollary

4.12. Finally, the results for the bags can be joined using dynamic programming along the clique separator decomposition.[5]

Using further advanced arguments (which are beyond the scope of this article) this result can be strengthened in two directions. First, by constructing isomorphism-invariant decompositions of k-basic graphs, one can design an fpt isomorphism test for graphs of tree-width k. This was first achieved by Lokshtanov, Pilipczuk, Pilipczuk and Saurabh [23] who designed an isomorphism test running in time $2^{O(k^5 \log k)} n^5$. This was later improved to $2^{k \cdot \text{polylog}(k)} n^3$ [24] using a more refined analysis of the decomposition as well as incorporating the group-theoretic machinery via Theorem 4.6.

Second, isomorphism of graphs of tree-width k can also be tested in time $n^{\text{polylog}(k)}$. Here, the crucial step is to speed up the dynamic programming procedure along the clique separator decomposition which was achieved by Wiebking [28].

5.1.3. Graphs excluding a topological subgraph

So far, we have seen isomorphism tests running in time $n^{\text{polylog}(k)}$ for k being the maximum degree, genus, or tree-width of the input graphs. All three algorithms follow a similar pattern of decomposing the input graphs into pieces that are t-CR-bounded after individualizing a constant number of vertices (where depends linearly on k). This naturally raises the question of a common generalization.

A graph H is a *topological subgraph* of a graph G if H can be obtained from G by deleting vertices and edges, and dissolving vertices of degree 2. Here, dissolving vertices of degree 2 means deleting the vertex and connecting its two neighbors by an edge. Equivalently, H is a topological subgraph of a graph G if there is a subgraph G' of G that is a subdivision of H, i.e., each edge of H is replaced by a path of arbitrary length. A graph G *excludes H as a topological subgraph* if G has no topological subgraph isomorphic to H. Observe that every topological subgraph of G is also a minor of G. Hence, if G excludes a graph H as a minor, then G also excludes H as a topological subgraph.

Note that, if H is a topological subgraph of a graph G, then the maximum degree of H is at most the maximum degree of G. Hence, graphs of maximum degree d exclude K_{d+2}, the complete graph on $d+2$ vertices, as a topological subgraph. Also, it is known that graphs of tree-width at most k exclude K_{k+2} as a minor, and graphs of genus at most g exclude K_{g+7} as a minor. So excluding a clique of size h as a topological subgraph is a common generalization of the parameterization by maximum degree, genus and tree-width. In particular, the following theorem generalizes the results discussed so far in this section.

Theorem 5.9 (Neuen [30]). *The Graph Isomorphism Problem for graphs excluding K_h as a topological subgraph can be solved in time $n^{\text{polylog}(h)}$.*

As indicated above, the general strategy to obtain an isomorphism test for graphs excluding K_h as a topological subgraph remains the same, i.e., we decompose the graph in an isomorphism-invariant way into pieces that are t-CR-bounded after individualizing a constant number of vertices. However, the technical details become significantly more intricate. In the following, we only discuss the key ideas.

Similar to the case of graphs of bounded tree-width, the main challenge is to find an appropriate isomorphism-invariant decomposition. For graphs of bounded genus and for graphs of bounded tree-width, we could rely on existing results to decompose the graphs, and the key remaining task was to prove that the pieces of the decomposition are indeed t-CR-bounded after fixing a constant number of vertices. Unfortunately, for graphs excluding K_h as a topological subgraph, similar decomposition results do not seem to be available.

Here, the key idea is to use the t-CR procedure itself as a way of decomposing the input graphs. For example, we can interpret the proof of Lemma 5.8 as stating that, in a graph of tree-width at most k, the k-closure $\text{cl}_k^G(v)$ of a vertex v can only stop to "grow" at a clique separator in the k-improved graph. This way, we can use the t-CR procedure itself to identify suitable separators. It turns out that the same still holds true for graphs excluding K_h as a topological subgraph.

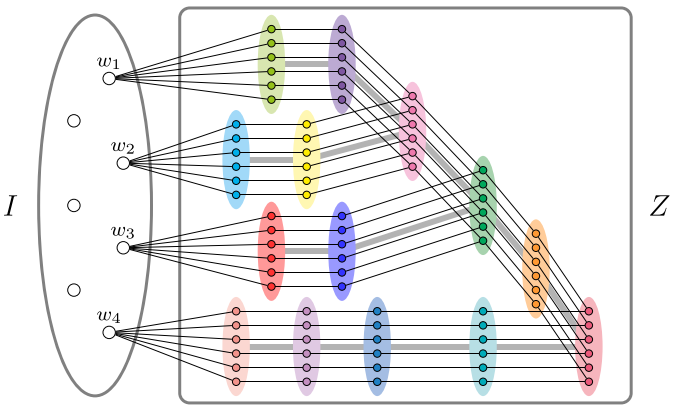

Fig. 8. The figure shows the construction of a topological subgraph K_h for $h = 4$ in the proof sketch of Lemma 5.10. We construct $\binom{h}{2} = 6$ vertex-disjoint subgraphs each of which contains exactly one vertex from each color class on the right. This allows us to build a topological subgraph K_4 by connecting all pairs $w_i w_j$, $i \neq j$, with vertex-disjoint paths. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

Lemma 5.10 ([29, Theorem 4.1]). *Let $t \geq 3h^3$. Let G be a graph that excludes K_h as a topological subgraph. Also let $X \subseteq V(G)$ and let Z be the vertex set of a connected component of $G - \text{cl}_t^G(X)$. Then $|N(Z)| < h$.*

Proof (Sketch). The proof is similar in spirit to the proof of Lemma 5.2; see also Fig. 8 for a visualization.

Let λ^* denote the vertex-coloring obtained from individualizing X, i.e., $\lambda^*(v) := (v, 1)$ for $v \in X$ and $\lambda^*(v) := (0, 0)$ for $v \in V(G) \setminus X$. Let $\chi := \chi_{t\text{-CR}}(G, \lambda^*)$ denote the t-CR-stable coloring with respect to the input graph (G, λ^*).

Let $I := \text{cl}_t^G(X)$. Also let Z be a connected component of $G - I$. Now let $C_I := \chi(I)$ denote the set of colors appearing in I, and let $C_Z := \chi(Z)$. Consider the graph H with vertex set $V(H) := C_I \cup C_Z$ and

$$E(H) := \{c_1 c_2 \mid \exists v_1, v_2 \in V(G) \colon \chi(v_1) = c_1, \chi(v_2) = c_2, v_1 v_2 \in E(G)\}.$$

Suppose towards a contradiction that $|N(Z)| \geq h$. Let us fix h distinct $w_1, \dots, w_h \in N_G(Z)$. Also let us fix h colors $c_1, \dots, c_h \in C_Z$ such that w_i has a neighbor of color c_i for every $i \in \{1, \dots, h\}$. Moreover, let T' be a subtree of a spanning tree of $H[C_Z]$ with leaves exactly in $\{c_1, \dots, c_h\}$.

Now, the key step of the proof is to construct $\binom{h}{2}$ many vertex-disjoint subgraphs of G each of which visits all of the color classes present in T' (see Fig. 8). Here, we exploit the fact that all color classes represented by nodes in T' are sufficiently large, i.e., they have size at least $t \geq 3h^3$. Given these subgraphs, we immediately obtain a topological subgraph isomorphic to K_h. □

The previous lemma allows us to identify separators in the input by setting $X = \{v\}$ for some suitable choices of v. Unfortunately, this is not yet sufficient to obtain an isomorphism-invariant tree decomposition similar to Theorem 5.6. The issue is that different choices of $X = \{v\}$ lead to different separators that may overlap in an uncontrolled way. To circumvent this issue, we would like the set $\text{cl}_t^G(X)$ not to depend on our choice of the starting vertex v. Maybe surprisingly, this can indeed be achieved assuming $t = \Omega(h^5)$. More concretely, it can be shown that after running the 3-WL algorithm (see Section 3), there is a vertex

[6] Technically, this requires solving a more general isomorphism problem for k-basic graphs; we omit the details here and refer to [24,29] instead.

color class $X \subseteq V(G)$ such that $\mathrm{cl}_t^G(v) = \mathrm{cl}_t^G(w)$ for all $v, w \in X$. This implies that $D := \mathrm{cl}_t^G(v)$ does not depend on the choice of v and is indeed isomorphism-invariant. This allows us to choose D as the root bag of the decomposition and decompose the graph recursively. Note that, by definition, each piece of the resulting decomposition is t-CR-bounded after individualizing a single vertex. So we can build an isomorphism test running in time $n^{\mathrm{polylog}(h)}$ similar to before.

5.2. FPT algorithms

Having presented isomorphism tests running in time $n^{\mathrm{polylog}(k)}$ for various graph parameters k, let us now turn to the design of fpt algorithms. We already indicated in Section 5.1.2 how to obtain fpt isomorphism tests parameterized by the tree-width of the input graphs [23,24]. In the following, we focus on parameterization by the genus and the size of an excluded minor.

5.2.1. Graphs of bounded genus

We start by considering parameterization by the genus of the input graphs. In 2015, Kawarabayashi [62] proposed, for every integer g, a linear-time isomorphism test for graphs of genus at most g. Unfortunately, even though a central subroutine has been simplified recently [63], Kawarabayashi's algorithm remains very complicated and technical and it is also not clear that the algorithm can be realized in a strongly uniform manner (i.e., [62] only states the existence of a non-uniform fpt isomorphism test). Instead, we present an fpt isomorphism test running in time $2^{O(g^4 \log g)} n^{O(1)}$ for graphs of genus at most g [25].

The idea for the algorithm is to build on the approach presented in the previous subsection and reduce it to the isomorphism problem for t-CR-bounded graphs. Unfortunately, no fpt isomorphism test parameterized by the minimal t, for which the input graphs are t-CR-bounded, is currently available (in fact, it is a long-standing open question whether isomorphism testing is fpt when parameterized by the maximum degree [17,18]). Hence, contrary to the previous subsection, we need to reduce it to the isomorphism problem for t-CR-bounded graphs where t is a fixed constant.

The key to this reduction is an adaptation of the argument underlying Lemma 5.10. Recall that graphs of genus at most g exclude $K_{3,4g+3}$ as a minor. In fact, as in Section 5.1.1, this is the only property the algorithm is going to exploit, i.e., the algorithm actually works for all graphs excluding $K_{3,h}$ as a minor. More concretely, as in Lemma 5.10, we aim to prove that the t-closure can only stop to "grow" at a separator of size strictly less than h. To obtain a minor isomorphic to $K_{3,h}$, we again choose h vertices $w_1, \dots, w_h$ in the set I. Now, the main observation is that, in order to construct a minor $K_{3,h}$, we only require three connected subgraphs within the set $\chi^{-1}(C_Z)$ each of which visits every color class in C_Z. So it seems reasonable that it suffices for all color classes in C_Z to have size larger than some fixed constant. This can in fact be achieved, but with the small caveat that we require the coloring χ not to be refined by 2-WL (instead of the weaker Color Refinement algorithm). Here, we say that a vertex-coloring χ is not refined by 2-WL if the coloring computed by 2-WL, restricted to the diagonal entries of the form (v, v), does not refine χ, where we naturally identify the tuple (v, v) with the vertex v.

Lemma 5.11 ([25, Lemma 5.4]). *Let (G, χ) be a connected vertex-colored graph such that χ is not refined by 2-WL, and $|\chi^{-1}(c)| \geq 3$ for all $c \in \mathrm{im}(\chi)$. Then there are three vertex-disjoint subgraphs H_1, H_2, H_3 of G such that $V(H_r) \cap \chi^{-1}(c) \neq \emptyset$ for all $c \in \mathrm{im}(\chi)$, and all $r \in \{1, 2, 3\}$.*

We note that switching to the 2-WL algorithm does not pose any significant challenges as all relevant results can be extended. Implementing this idea leads to the following result.

Theorem 5.12 (Neuen [25]). *The Graph Isomorphism Problem for graphs excluding $K_{3,h}$ as a minor can be solved in time $2^{O(h^4 \log h)} n^{O(1)}$.*

In particular, we obtain an fpt isomorphism test for graphs of bounded genus.

Still, the result has some clear drawbacks. Maybe most significantly, due to the usage of the group-theoretic machinery, the polynomial dependence on n is rather large which makes the algorithm impractical. Also, using Kawarabayashi's algorithm instead leads to some problems due to its complicated nature and high dependence on the genus g. For this reason, let us propose a different approach that raises some interesting open questions. This approach is partly motivated by practical isomorphism solvers such as Nauty/Traces [40,43] and Bliss [41,42] which rely on the individualization/refinement paradigm that uses individualization of vertices in combination with the Color Refinement algorithm (or similar refinement algorithms) to test isomorphism. In a nutshell, the idea is to repeatedly individualize a vertex, that does not yet have a unique color, and perform the Color Refinement algorithm until the obtained coloring is discrete. At this point, isomorphism is easy to decide.

For a graph G and vertices $v_1, \dots, v_\ell$, we denote by $\chi_{(\infty)}(G; v_1, \dots, v_k)$ the coloring obtained from the Coloring Refinement algorithm after individualizing the vertices $v_1, \dots, v_k$.

Definition 5.13. Let G be a graph. An *individualization sequence for G* is a tuple $(v_1, \dots, v_k)$ such that, for every $i \in [k]$, it holds that $|[v_i]_{\lambda_i}| > 1$ where $\lambda_i := \chi_{(\infty)}(G; v_1, \dots, v_{i-1})$.

An individualization sequence is *maximal* if λ_k is discrete, i.e., the individualization sequence cannot be extended by another element.

The *cost* of an individualization sequence $(v_1, \dots, v_k)$ is

$$\mathrm{cost}(v_1, \dots, v_k) := \prod_{i \in [k]} |[v_i]_{\lambda_i}|.$$

Observe that, if $(v_1, \dots, v_k)$ is a maximal individualization sequence for G, then isomorphism between G and any other graph can be tested in time $O(\mathrm{cost}(v_1, \dots, v_k) \cdot (n+m) \log n)$ where the second part of the running time is the time required to execute the Color Refinement algorithm.

Question 5.14. Does every 3-connected graph of genus at most g have a maximal individualization sequence of cost at most $f(g) \cdot n$ (for some computable function f)?

We remark that every planar graph has an individualization sequence with cost at most $20n$ [48]. Actually, we may even ask for a stronger statement. Observe that, in order to keep the cost small, it generally seems preferable to individualize vertices in small color classes. Since every 3-connected graph of genus at most g is $(4g+2)$-CR-bounded (after individualizing one vertex of small degree), the existence of a color class of small size is always guaranteed.

Question 5.15. Is there a function f such that, for every 3-connected graph of genus at most g, and every individualization sequence $(v_1, \dots, v_k)$ for G, it holds that $k \leq f(g)$?

Note that we ask whether *every* individualization sequence is short. In such a situation, we could simply focus on choosing the smallest color class for individualizing a vertex to obtain a small cost. This question is also interesting from a practical point of view, since most isomorphism solvers choose small color classes for individualization in an effort to keep the cost low. Hence, a positive answer would imply that many practical solvers run in fpt time on 3-connected graphs parameterized by the genus.

5.2.2. Graphs excluding a minor

Given fpt isomorphism tests parameterized by the genus and the tree-width of the input graphs, the next natural step is to target an fpt algorithm parameterized by the size of a forbidden minor. This was recently achieved by Lokshtanov, Pilipczuk, Pilipczuk and Saurabh [26] with the small caveat that their algorithm is not strongly uniform.

Theorem 5.16 (Lokshtanov, Pilipczuk, Pilipczuk, Saurabh [26]). *There is an algorithm that, given a graph H and two H-minor-free*

graphs G_1 and G_2, decides in time $f(H) \cdot n^{O(1)}$ whether G_1 is isomorphic to G_2 for some function f.

Observe that the theorem does not guarantee the function f to be computable which means that it does not provide a (strongly uniform) fpt algorithm.

The details of the isomorphism test from [26] are quite complicated and we only provide a rough sketch here. The algorithm follows again a decomposition approach. More concretely, the first part of the algorithm reduces isomorphism testing to graphs that are *unbreakable*. A graph G is (q,k)-*unbreakable* if for every separation (A,B) of G of order at most k it holds that $|A| \leq q$ or $|B| \leq q$. The first part essentially splits the input graphs into pieces that are $(q(i),i)$-unbreakable for every $i \leq k$, where k and the function q are chosen appropriately (and depend only on H). We remark that a decomposition into unbreakable parts was known before (see, e.g., [64]), but the main challenge is to make this decomposition isomorphism-invariant so that it can be used for isomorphism testing. While this goal is not quite achieved in [26], it is still possible to provide a reduction to the unbreakable case by employing further tricks.

Given an unbreakable graph G, it is shown that the vertex set can be partitioned into two parts $V_{\text{tw}} \uplus V_{\text{genus}}$ so that $G[V_{\text{tw}}]$ has bounded tree-width and $G[V_{\text{genus}}]$ is fairly "rigid" in the sense that we can compute an fpt-sized set of "candidate isomorphisms" such that every actual isomorphism (to the corresponding subgraph of the second input graph) is contained in this set. This allows us to test isomorphism by iterating over all "candidate isomorphisms", and testing whether it extends to the entire graph by exploiting that $G[V_{\text{tw}}]$ has bounded tree-width (which allows the use of existing fpt algorithms parameterized by tree-width).

Of course, the most natural open question is whether it is possible to obtain a strongly uniform fpt algorithm. Towards this end, it may also be interesting to consider intermediate problems. Let G be a graph. The *planar deletion number* is the minimal number of vertices that need to be removed from G in order to make it planar. It is not difficult to see that graphs of planar deletion number at most t exclude K_{t+5} as a minor. Hence, the theorem above provides a weakly uniform fpt isomorphism test parameterized by the planar deletion number. To the best of the author's knowledge, no other fpt algorithms are known.

Question 5.17. Is there a simple (strongly uniform) fpt isomorphism test parameterized by the planar deletion number?

6. Dense graphs

In the last section, we discussed isomorphism tests for parametrization by several "sparse" graph parameters. More concretely, all graph parameters k considered in the last section share the property that the considered graphs are d-degenerate where d only depends on the parameter k in question. In particular, this implies that, as long as the parameter k is bounded, we are only considering graphs with a linear number of edges.

In this section, we turn our attention to "dense" graph parameters, i.e., graph parameters which may be bounded even if the number of edges is quadratic in the number of vertices. For example, a common feature of these parameters (with the exception of eigenvalue multiplicity) is that they assign a constant value to all complete graphs. Note that isomorphism testing for complete graphs is trivial, and in general there does not seem to be any a priori hurdle to obtaining similar results as in the previous section for "dense" graph parameters. However, up to this point, far fewer results are available in the dense setting and many questions remain open.

6.1. Rank-width and beyond

Two of the most important graph parameters in the dense setting are *clique-width* and *rank-width*. Since both parameters are functionally equivalent (i.e., both parameters are bounded by a function of the other) [65], we focus only on rank-width in the following. The parameter rank-width can be viewed as a "dense analogue" of tree-width which is one of the most important and well-studied graph parameters in parameterized complexity and structural graph theory.

We start by giving the definition of rank-width. Let G be a graph. For $X, Y \subseteq V(G)$ we define $A_G(X,Y) \in \mathbb{F}_2^{X \times Y}$ to be the matrix defined via $(A_G(X,Y))_{v,w} = 1$ if $vw \in E(G)$, and $(A_G(X,Y))_{v,w} = 0$ otherwise. Observe that $A_G(X,Y)$ is obtained from the adjacency matrix of G by only keeping the rows indexed by vertices from X and the columns indexed by vertices from Y. We define $\rho_G(X) := \text{rk}_2(A_G(X, V(G) \setminus X))$ where $\text{rk}_2(A)$ denotes the $\mathbb{F}_2$-rank of a matrix A.

A *rank decomposition* of a graph G is a pair (T,γ) where T is a rooted binary tree and $\gamma : V(T) \to 2^{V(G)}$ such that the following holds:

(R.1) $\gamma(r) = V(G)$ where r is the root of T,
(R.2) $\gamma(t) = \gamma(s_1) \cup \gamma(s_2)$ and $\gamma(s_1) \cap \gamma(s_2) = \emptyset$ for every inner node $t \in V(T)$ with children s_1 and s_2, and
(R.3) $|\gamma(t)| = 1$ for every leaf t of T.

Note that (R.1)–(R.3) imply that the leaves of T are in one-to-one correspondence with the vertices of G. The *width* of (T,γ) is $\max_{t \in V(T)} \rho_G(\gamma(t))$. The *rank-width* of G, denoted by $\text{rw}(G)$, is the minimum width of a rank decomposition of G. An example of a rank decomposition is given in Fig. 9. We note that $\text{rw}(G) \leq \text{tw}(G) + 1$ for every graph G [66].

The first polynomial-time isomorphism test for graphs of bounded rank-width k was provided by Grohe and Schweitzer [15] and runs in time $n^{f(k)}$ for some *non-elementary* function f. This was later improved by the following result which shows that the WL algorithm of dimension $O(k)$ serves as a complete isomorphism test.

Theorem 6.1 (Grohe, Neuen [67]). *Let G be a graph of rank-width at most k. Then the WL dimension of G is at most $3k+4$.*

In particular, the Graph Isomorphism Problem for graphs of rank-width at most k can be solved in time $n^{O(k)}$.

No faster isomorphism tests are known so far. In particular, in contrast to parameterization by tree-width, neither an fpt isomorphism test nor an isomorphism test running in time $n^{\text{polylog}(k)}$ is known.

Question 6.2. Is there an fpt isomorphism test parameterized by rank-width? Is there an isomorphism test running in time $n^{\text{polylog}(k)}$ for graphs of rank-width at most k?

A first step towards answering the first question has recently been taken in [68] by obtaining a non-uniform fpt isomorphism test parameterized by *rank-depth* [69]. Intuitively speaking, for the rank-depth, we not only restrict the width of the decomposition, but also the depth of the underlying tree T. In order to still be able to have graphs of arbitrary size (while bounding the depth of the tree), we need to allow nodes of arbitrarily high degree in the tree T.

A *generalized rank decomposition* of a graph G is a pair (T,γ) where T is a rooted tree and $\gamma : V(T) \to 2^{V(G)}$ such that the following holds:

(R.1*) $\gamma(r) = V(G)$ where r is the root of T,
(R.2*) $\gamma(t) = \gamma(s_1) \uplus \cdots \uplus \gamma(s_d)$ for every inner node $t \in V(T)$ with children $s_1, \dots, s_d$, and
(R.3*) $|\gamma(t)| = 1$ for every leaf t of T.

The *width* of (T,γ) is $\max_{t \in V(T), S \subseteq N_+(t)} \rho_G\left(\bigcup_{s \in S} \gamma(s)\right)$ (i.e., S ranges over all subsets of children of every node t). The *rank-depth* of G is the minimum $k \geq 1$ such that there is a generalized rank decomposition (T,γ) of G of width at most k where T has depth at most k.

Theorem 6.3 (Ohlmann, Pilipczuk, Przybyszewski, Torunczyk [68]). *Let C be a class of graphs of bounded rank-depth. Then the Graph Isomorphism Problem for C can be solved in time $O(n^2)$.*

Let us stress again that this is a non-uniform fpt isomorphism test since for every k there is a separate algorithm testing isomorphism on all graphs of rank-depth at most k. Interestingly, the algorithm takes a logical approach to the problem. The authors show that, for every class of

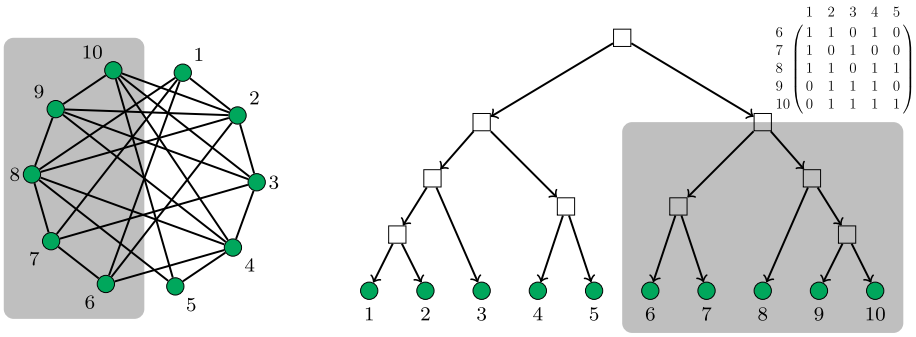

Fig. 9. A graph G on the left and a potential rank decomposition (T, γ) of G of width 3 on the right. The function γ is completely specified by the bijection between the leaves of T and the vertices of G. As an example, the matrix $A_G(\{6,7,8,9,10\}, \{1,2,3,4,5\})$ is provided. Note that $\rho_G(\{6,7,8,9,10\}) = 3$.

graphs $\mathcal{C}$ of bounded rank-depth, there exists a mapping $\mathcal{A}$ that translates each $G \in \mathcal{C}$ into a graph $\mathcal{A}(G)$ of bounded tree-depth while preserving isomorphisms. After that, we can rely on existing fpt isomorphism tests parameterized by the tree-depth of the input graphs [22]. The existence of the sought mapping $\mathcal{A}$ is proved via model-theoretic tools and in particular builds on the Compactness Theorem for first-order logic. More concretely, the definition of $\mathcal{A}$ relies on the existence of certain first-order formulas for which it is unclear whether they can be computed given a number k. This makes the algorithm non-uniform.

Of course, it would also be nice to further generalize Theorem 6.1. In the sparse setting, graphs with forbidden minors form a natural generalization of graphs of bounded tree-width, and there exists a corresponding notion of a *vertex-minor* in the dense setting. For a graph G and a vertex $v \in V(G)$ we define $G * v$ as the graph obtained from G by complementing all edges between vertices in the neighborhood of v. This operation is referred to as a *local complementation*. A graph H is a *vertex-minor* of G if H can be obtained from G by deleting vertices and performing local complementations. We remark that if H is a vertex-minor of G, then $\mathrm{rw}(H) \leq \mathrm{rw}(G)$. Also, it is known that graphs of rank-width at most k can be characterized by a finite list of forbidden vertex-minors [70]. A positive answer to the next question would thus provide a broad generalization of Theorem 6.1.

Question 6.4. Is there, for every fixed graph H, a polynomial-time algorithm for testing isomorphism for all graphs excluding H as a vertex-minor?

6.2. Intersection graphs

Another way of obtaining dense classes of graphs is to consider *intersection graphs* of certain objects. Two well-known examples are *interval graphs* and *unit disk* graphs. A graph G is an *interval graph* if each vertex can be identified with a closed interval on the real line such that two vertices are adjacent if and only if the corresponding intervals intersect. Unit disk graphs are defined in the same way, but vertices are identified with unit disks in the plane. We remark that isomorphism of interval graphs can be tested in polynomial time; for unit disk graphs the complexity of isomorphism testing is open.

We briefly discuss a series of recent works that eventually obtained an fpt isomorphism test for chordal graphs parameterized by their *leafage*. A graph G is chordal if every cycle of length at least 4 in G has a chord. Alternatively, chordal graphs can also be described as intersection graphs of subtrees of an arbitrary fixed tree T. It is not difficult to show that isomorphism testing of chordal graphs is GI-complete (i.e., as difficult as the general isomorphism problem). Hence, it makes sense to consider parameterizations of chordal graphs. Looking at the definition via intersection graphs, the following parameterization is natural. The *leafage* of a chordal graph G is the minimum number ℓ such that G can be represented as an intersection graph of subtrees of a tree T with ℓ leaves. It was proved independently in [71,72] that isomorphism testing of chordal graphs is fpt parameterized by the leafage.

Theorem 6.5 (Arvind, Nedela, Ponomarenko, Zeman [71]; Çagirici, Hlinený [72]). *The Graph Isomorphism Problem for chordal graphs is fpt parameterized by the leafage of the input graphs.*

6.3. Eigenvalue multiplicity

Next, let us briefly discuss parameterization by eigenvalue multiplicity. In contrast to the other parameters considered in this work, which examine combinatorial features of the input graphs, the eigenvalue multiplicity is an algebraic feature of the adjacency matrix.

Let G be a graph and let $A_G \in \{0,1\}^{n \times n}$ denote its adjacency matrix (where n denotes the number of vertices). We say that $\lambda \in \mathbb{R}$ is an *eigenvalue* of G if it is an eigenvalue of its adjacency matrix, i.e., there is a non-zero vector $v \in \mathbb{R}^n$ such that $A_G v = \lambda v$. The *multiplicity* of an eigenvalue λ is the dimension of its eigenspace $W = \{v \in \mathbb{R}^n \mid A_G v = \lambda v\}$. Let $\lambda_1, \dots, \lambda_\ell$ denote the eigenvalues of A_G, and let m_i denote the multiplicity of λ_i. Since A_G is a symmetric matrix we obtain that $\sum_{i=1,\dots,\ell} m_i = n$. The *eigenvalue multiplicity* of G is the maximal multiplicity among its eigenvalues.

The first polynomial-time isomorphism test for graphs of bounded eigenvalue multiplicity was obtained by Babai, Grigoryev and Mount [8] and runs in time $n^{O(k)}$. This was later improved by Evdokimov and Ponomarenko to obtain one of the first fpt algorithms in the context of isomorphism testing.

Theorem 6.6 (Evdokimov, Ponomarenko [19]; Arvind, Rattan [20]). *The Graph Isomorphism Problem for graphs of eigenvalue multiplicity at most k can be solved in time $2^{O(k \log k)} n^{O(1)}$.*

We note that [19] only obtains the running time bound $2^{O(k \log k)} n^{O(1)}$ under the Classification of Finite Simple Groups (CFSG), an extremely deep classification result from group theory. Already in their paper, Evdokimov and Ponomarenko provide a weaker runtime bound of $2^{O(k^2/\log k)} n^{O(1)}$ which avoids the use of CFSG. An alternative algorithm by Arvind and Rattan [20] completely avoids the dependence on CFSG.

6.4. Twin-width and tournaments

We close this section by discussing the parameterization of isomorphism testing by twin-width. Twin-width is a graph parameter introduced by Bonnet et. al [73] that has gained significant attention in both structural graph theory and parameterized complexity (see, e.g., [73–78]).

We start by describing its definition. Let G be a graph. Two vertices $v, w \in V(G)$ are *twins* if $N_G(v) \setminus \{v, w\} = N_G(w) \setminus \{v, w\}$, i.e., v, w have the same neighbors outside of $\{v, w\}$. If two vertices in a graph are twins, then it is often possible to "merge" the two vertices into a single vertex

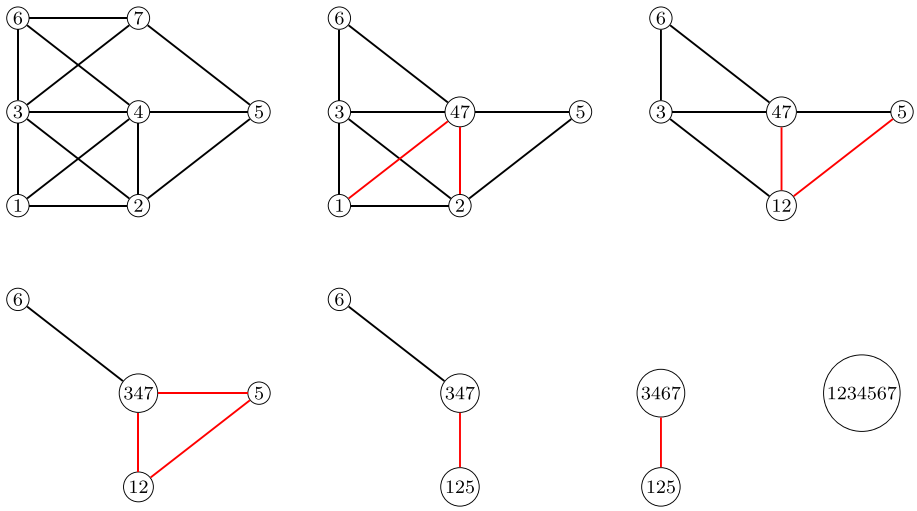

Fig. 10. A 2-contraction sequence of the graph G in the top-right corner. As a result, G has twin-width at most 2.

without affecting the problem at hand. The key idea underlying the parameter twin-width is to allow merging of vertices even if they are not twins, but to introduce a *red edge* (which signals an error) to all vertices that are connected to exactly one of the vertices v, w. Now, the goal is to gradually merge more and more vertices until eventually we arrive at a graph with only a single vertex. Throughout this process, we aim to control the amount of red edges that are introduced.

A *trigraph* is a triple $G = (V, E, R)$ where V is a finite vertex set and $E, R \subseteq \{vw \mid v, w \in V\}$ are disjoint sets of unordered pairs. We refer to the elements of E and R as the *edges* and *red edges* of G, respectively. Consistent with the previous notation, we use $V(G)$, $E(G)$ and $R(G)$ to denote the vertices, edges and red edges of a trigraph G. A *trigraph* $G = (V, E, R)$ has *red degree at most* d if $N^R(v) := \{w \in V \mid vw \in R\}$ has size at most d for every $v \in V$.

Let $G = (V, E, R)$ be a trigraph. For two vertices $v, w \in V(G)$ we define the trigraph $G/v, w = (V', E', R')$ obtained by identifying v, w into a fresh vertex z (i.e., $V' = (V \setminus \{v, w\}) \cup \{z\}$) such that all edges and red edges not incident to v, w remain unchanged and the following edges incident to z are present:

- $zx \in E'$ if and only if $vx \in E$ and $wx \in E$,
- $zx \notin E' \cup R'$ if and only if $vx \notin E \cup R$ and $wx \notin E \cup R$, and
- $zx \in R$ otherwise.

We say that $G/v, w$ is a *contraction* of G. A trigraph G has a d-*contraction sequence* if there exist $G = G_n, \dots, G_1$ such that G_i is a contraction of G_{i+1} for every $i \in [n-1]$, and G_i has red degree at most d for every $i \in [n]$, and $|V(G_1)| = 1$. A visualization is given in Fig. 10. The *twin-width* of a trigraph G is the minimum d such that G has a d-contraction sequence. The twin-width of a graph G is the twin-width of the corresponding trigraph $(V(G), E(G), \emptyset)$ with an empty set of red edges.

Now, let us study the complexity of the Graph Isomorphism Problem parameterized by the twin-width of the input graph. Unfortunately, it turns out that even testing the isomorphism of graphs with twin-width 4 is as hard as the general isomorphism problem.

Theorem 6.7. *The Graph Isomorphism Problem for graphs of twin-width at most* 4 *is GI-complete.*

Proof. Let G, H be two graphs. We construct G' from G by replacing every edge with a path of length $\lceil 2 \log n \rceil$; the graph H' is obtained in the same way from H. Then $G \cong H$ if and only if $G' \cong H'$. Also, G' and H' have twin-width at most 4 [79]. □

This means parameterization by twin-width is not interesting for general graphs in the context of isomorphism testing. However, it turns out that an fpt isomorphism test can be obtained if we restrict ourselves to the special case of testing isomorphism of tournaments. A *tournament* is a directed graph T where for each pair of distinct vertices $v, w \in V(T)$ exactly one of the pairs (v, w) and (w, v) is an edge.

Tournament Isomorphism Problem (TI)
Input: Two tournaments T_1, T_2.
Question: Decide whether $T_1 \cong T_2$.

It is not difficult to obtain a polynomial-time many-one reduction from TI to GI. In the other direction, no polynomial-time reduction is known and TI is generally considered to be a simpler problem. For example, an algorithm solving TI in time $n^{O(\log n)}$ was already obtained in the 1980's [16]; it took over three more decades to find a quasipolynomial-time isomorphism test for all graphs [2]. Also, the automorphism group of each tournament is solvable which allows for the use of various subroutines from computational group theory. Despite the availability of such tools, no improvement over the $n^{O(\log n)}$ time algorithm from [16] is known. Hence, it is natural to consider parameterized versions of TI. The notion of twin-width naturally carries over to directed graphs, and hence to tournaments (the structure of tournaments of bounded twin-width is studied in [78]).

Theorem 6.8 (Grohe, Neuen [80]). *The isomorphism problem for tournaments of twin-width at most k can be solved in time $k^{O(\log k)} n^{O(1)}$.*

Note that, in light of Theorem 6.7, this result can be viewed as another piece of evidence that TI may be simpler than GI. Also, we remark that on tournaments, twin-width is functionally smaller than many other natural graph parameters, including clique-width, directed tree-width, directed path-width and cut-width. As a result, TI parameterized by any of these parameters is also fpt.

Naturally, it would be nice to further generalize Theorem 6.8. Here, an interesting parameter may be the VC dimension of the set system defined by the out-neighbors of all vertices $v \in V(T)$. Let $\mathcal{F}$ be a family of subsets of some finite universe V. A set $S \subseteq V$ is *shattered by* $\mathcal{F}$ if for every $T \subseteq S$ there is some $F \in \mathcal{F}$ such that $T = S \cap F$. The *VC dimension* of $\mathcal{F}$ is the largest integer d such that some d-element subset of V is shattered by $\mathcal{F}$ (see, e.g., [81]). We define the *VC dimension* of a tournament T to be the VC dimension of the set system $\mathcal{F} = \{N^+(v) \mid v \in V(T)\}$ where $N^+(v)$ denotes the outgoing neighbors of v. We note that the VC dimension of a tournament is bounded by its twin-width, but not the other way around [78]. This leads to the following question.

Question 6.9. Can isomorphism for tournaments of bounded VC dimension be tested in polynomial time?

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher (Eds.), Proceedings of a Symposium on the Complexity of Computer Computations, Held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA, The IBM Research Symposia Series, Plenum Press, New York, 1972, pp. 85–103, https://doi.org/10.1007/978-1-4684-2001-2_9.

[2] L. Babai, Graph isomorphism in quasipolynomial time [extended abstract], in: D. Wichs, Y. Mansour (Eds.), Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016, ACM, 2016, pp. 684–697, https://doi.org/10.1145/2897518.2897542

[3] L. Babai, W.M. Kantor, E.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, IEEE Computer Society, 1983, pp. 162–171, https://doi.org/10.1109/SFCS.1983.10

[4] L.C. Ray, R.A. Kirsch, Finding chemical records by digital computers, Science 126 (3278) (1957) 814–819, https://doi.org/10.1126/science.126.3278.814

[5] S.H. Unger, GIT - a heuristic program for testing pairs of directed line graphs for isomorphism, Commun. ACM 7 (1) (1964) 26–34, https://doi.org/10.1145/363872.363899

[6] J.E. Hopcroft, R.E. Tarjan, Isomorphism of planar graphs, in: R.E. Miller, J.W. Thatcher (Eds.), Proceedings of a Symposium on the Complexity of Computer Computations, Held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA, The IBM Research Symposia Series, Plenum Press, New York, 1972, pp. 131–152, https://doi.org/10.1007/978-1-4684-2001-2_13.

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

[8] L. Babai, D.Y. Grigoryev, D.M. Mount, Isomorphism of graphs with bounded eigenvalue multiplicity, in: H.R. Lewis, B.B. Simons, W.A. Burkhard, L.H. Landweber (Eds.), Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA, ACM, 1982, pp. 310–324, https://doi.org/10.1145/800070.802206

[9] G.L. Miller, Isomorphism testing for graphs of bounded genus, in: R.E. Miller, S. Ginsburg, W.A. Burkhard, R.J. Lipton (Eds.), Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA, ACM, 1980, pp. 225–235, https://doi.org/10.1145/800141.804670

[10] G.L. Miller, Isomorphism of k-contractible graphs. A generalization of bounded valence and bounded genus, Inf. Control. 56 (1/2) (1983) 1–20, https://doi.org/10.1016/S0019-9958(83)80047-3

[11] G.L. Miller, Isomorphism of graphs which are pairwise k-separable, Inf. Control. 56 (1/2) (1983) 21–33, https://doi.org/10.1016/S0019-9958(83)80048-5

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

[13] I.N. Ponomarenko, The isomorphism problem for classes of graphs closed under contraction, J. Sov. Math. 55 (2) (1991) 1621–1643, https://doi.org/10.1007/BF01098279

[14] M. Grohe, D. Marx, Structure theorem and isomorphism test for graphs with excluded topological subgraphs, SIAM J. Comput. 44 (1) (2015) 114–159, https://doi.org/10.1137/120892234

[15] M. Grohe, P. Schweitzer, Isomorphism testing for graphs of bounded rank width, in: V. Guruswami (Ed.), IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015, IEEE Computer Society, 2015, pp. 1010–1029, https://doi.org/10.1109/FOCS.2015.66

[16] L. Babai, E.M. Luks, Canonical labeling of graphs, in: D.S. Johnson, R. Fagin, M.L. Fredman, D. Harel, R.M. Karp, N.A. Lynch, C.H. Papadimitriou, R.L. Rivest, W.L. Ruzzo, J.I. Seiferas (Eds.), Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA, ACM, 1983, pp. 171–183, https://doi.org/10.1145/800061.808746

[17] R.G. Downey, M.R. Fellows, Parameterized Complexity, Monographs in Computer Science, Springer, 1999, https://doi.org/10.1007/978-1-4612-0515-9

[18] R.G. Downey, M.R. Fellows, Fundamentals of Parameterized Complexity, Texts in Computer Science, Springer, 2013, https://doi.org/10.1007/978-1-4471-5559-1

[19] S. Evdokimov, I.N. Ponomarenko, Isomorphism of coloured graphs with slowly increasing multiplicity of Jordan blocks, Comb. 19 (3) (1999) 321–333, https://doi.org/10.1007/S004930050059

[20] V. Arvind, G. Rattan, Faster FPT algorithm for graph isomorphism parameterized by eigenvalue multiplicity, CoRR arXiv:1408.3510, 2014.

[21] S. Kratsch, P. Schweitzer, Isomorphism for graphs of bounded feedback vertex set number, in: H. Kaplan (Ed.), Algorithm Theory - SWAT 2010, 12th Scandinavian Symposium and Workshops on Algorithm Theory, Bergen, Norway, June 21-23, 2010. Proceedings, Vol. 6139 of Lecture Notes in Computer Science, Springer, 2010, pp. 81–92, https://doi.org/10.1007/978-3-642-13731-0_9

[22] A. Bouland, A. Dawar, E. Kopczynski, On tractable parameterizations of graph isomorphism, in: D.M. Thilikos, G.J. Woeginger (Eds.), Parameterized and Exact Computation - 7th International Symposium, IPEC 2012, Ljubljana, Slovenia, September 12-14, 2012. Proceedings, Vol. 7535 of Lecture Notes in Computer Science, Springer, 2012, pp. 218–230, https://doi.org/10.1007/978-3-642-33293-7_21

[23] D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, S. Saurabh, Fixed-parameter tractable canonization and isomorphism test for graphs of bounded treewidth, SIAM J. Comput. 46 (1) (2017) 161–189, https://doi.org/10.1137/140999980

[24] M. Grohe, D. Neuen, P. Schweitzer, D. Wiebking, An improved isomorphism test for bounded-tree-width graphs, ACM Trans. Algorithms 16 (3) (2020) :34:1–:34:31, https://doi.org/10.1145/3382082

[25] D. Neuen, Isomorphism testing parameterized by genus and beyond, SIAM J. Discret. Math. 38 (1) (2024) 453–484, https://doi.org/10.1137/22M1514076

[26] D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, S. Saurabh, Fixed-parameter tractability of graph isomorphism in graphs with an excluded minor, in: S. Leonardi, A. Gupta (Eds.), STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022, ACM, 2022, pp. 914–923, https://doi.org/10.1145/3519935.3520076

[27] D. Neuen, Hypergraph isomorphism for groups with restricted composition factors, ACM Trans. Algorithms 18 (3) (2022) :27:1–:27:50, https://doi.org/10.1145/3527667

[28] D. Wiebking, Graph isomorphism in quasipolynomial time parameterized by treewidth, in: A. Czumaj, A. Dawar, E. Merelli (Eds.), 47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference), Vol. 168 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum Für Informatik, 2020, pp. :103:1–:103:16, https://doi.org/10.4230/LIPIcs.ICALP.2020.103

[29] M. Grohe, D. Neuen, D. Wiebking, Isomorphism testing for graphs excluding small minors, SIAM J. Comput. 52 (1) (2023) 238–272, https://doi.org/10.1137/21m1401930

[30] D. Neuen, Isomorphism testing for graphs excluding small topological subgraphs, in: J.S. Naor, N. Buchbinder (Eds.), Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022, SIAM, 2022, pp. 1411–1434, https://doi.org/10.1137/1.9781611977073.59

[31] L. Babai, Canonical form for graphs in quasipolynomial time: preliminary report, in: M. Charikar, E. Cohen (Eds.), Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019, ACM, 2019, pp. 1237–1246, https://doi.org/10.1145/3313276.3316356

[32] M. Grohe, P. Schweitzer, The graph isomorphism problem, Commun. ACM 63 (11) (2020) 128–134, https://doi.org/10.1145/3372123

[33] M. Grohe, D. Neuen, Recent advances on the graph isomorphism problem, In K.K. Dabrowski, M. Gadouleau, N. Georgiou, M. Johnson, G.B. Mertzios, D. Paulusma (Eds.), Surveys in Combinatorics 2021, Cambridge University Press, 2021, pp. 187–234, https://doi.org/10.1017/9781009036214.006, London Mathematical Society Lecture Note Series.

[34] J.J. Rotman, An Introduction to the Theory of Groups, fourth, Vol. 148 of Graduate Texts in Mathematics, Springer-Verlag, New York, 1995, https://doi.org/10.1007/978-1-4612-4176-8

[35] J.D. Dixon, B. Mortimer, Permutation Groups, Vol. 163 of Graduate Texts in Mathematics, Springer-Verlag, New York, 1996, https://doi.org/10.1007/978-1-4612-0731-3

[36] A. Seress, Permutation Group Algorithms of Cambridge Tracts in Mathematics, vol. 152, Cambridge University Press, Cambridge, 2003, https://doi.org/10.1017/CBO9780511546549

[37] J. Flum, M. Grohe, Parameterized Complexity Theory, Texts in Theoretical Computer Science. An EATCS Series, Springer, 2006, https://doi.org/10.1007/3-540-29953-X

[38] M. Cygan, F.V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, S. Saurabh, Parameterized Algorithms, Springer, 2015, https://doi.org/10.1007/978-3-319-21275-3

[39] A. Cardon, M. Crochemore, Partitioning a graph in $O(|A| \log_2 |V|)$, Theor. Comput. Sci. 19 (1982) 85–98, https://doi.org/10.1016/0304-3975(82)90016-0

[40] B.D. McKay, Practical graph isomorphism, Congr. Numer. 30 (1981) 45–87.

[41] T.A. Junttila, P. Kaski, Engineering an efficient canonical labeling tool for large and sparse graphs, in: Proceedings of the Nine Workshop on Algorithm Engineering and Experiments, ALENEX 2007, New Orleans, Louisiana, USA, January 6, 2007, SIAM, 2007, https://doi.org/10.1137/1.9781611972870.13

[42] T.A. Junttila, P. Kaski, Conflict propagation and component recursion for canonical labeling, in: A. Marchetti-Spaccamela, M. Segal (Eds.), Theory and Practice of Algorithms in (Computer) Systems - First International ICST Conference, TAPAS 2011, Rome, Italy, April 18-20, 2011. Proceedings, Vol. 6595 of Lecture Notes in Computer Science, Springer, 2011, pp. 151–162, https://doi.org/10.1007/978-3-642-19754-3_16

[43] B.D. McKay, A. Piperno, Practical graph isomorphism, II, J. Symb. Comput. 60 (2014) 94–112, https://doi.org/10.1016/J.JSC.2013.09.003

[44] M. Anders, P. Schweitzer, Engineering a fast probabilistic isomorphism test, in: M. Farach-Colton, S. Storandt (Eds.), Proceedings of the Symposium on Algorithm Engineering and Experiments, ALENEX 2021, Virtual Conference, January 10-11, 2021, SIAM, 2021, pp. 73–84, https://doi.org/10.1137/1.9781611976472.6

[45] B. Weisfeiler, A. Leman, The reduction of a graph to canonical form and the algebra which appears therein, NTI 2 (9) (1968) 12–16.

[46] N. Immerman, E. Lander, Describing graphs: a first-order approach to graph canonization, In A.L. Selman (Ed.), Complexity Theory Retrospective: in Honor of Juris

Hartmanis on the Occasion of His Sixtieth Birthday, July 5, 1988, Springer New York, New York, NY, 1990, pp. 59–81, https://doi.org/10.1007/978-1-4612-4478-3_5
[47] J. Cai, M. Fürer, N. Immerman, An optimal lower bound on the number of variables for graph identification, Comb. 12 (4) (1992) 389–410, https://doi.org/10.1007/BF01305232
[48] S. Kiefer, I. Ponomarenko, P. Schweitzer, The Weisfeiler-Leman dimension of planar graphs is at most 3, J. ACM 66 (6) (2019) 44:1–44:31, https://doi.org/10.1145/3333003
[49] M. Grohe, Descriptive Complexity, Canonisation, and Definable Graph Structure Theory, Vol. 47 of Lecture Notes in Logic, Cambridge University Press, 2017, https://doi.org/10.1017/9781139028868
[50] S. Kiefer, The Weisfeiler-Leman algorithm: an exploration of its power, ACM SIGLOG News 7 (3) (2020) 5–27, https://doi.org/10.1145/3436980.3436982
[51] C. Morris, Y. Lipman, H. Maron, B. Rieck, N.M. Kriege, M. Grohe, M. Fey, K. Borgwardt, Weisfeiler and Leman go machine learning: the story so far, J. Mach. Learn. Res. 24 (333) (2023) 1–59, http://jmlr.org/papers/v24/22-0240.html.
[52] M. Grohe, D. Neuen, P. Schweitzer, A faster isomorphism test for graphs of small degree, SIAM J. Comput. 52 (6) (2023) FOCS18–1–FOCS18–36. https://doi.org/10.1137/19m1245293
[53] D. Neuen, The Power of Algorithmic Approaches to the Graph Isomorphism Problem, (Ph.D. thesis), RWTH Aachen University, Aachen, Germany, 2019, https://doi.org/10.18154/RWTH-2020-00160
[54] M.W. Liebeck, On graphs whose full automorphism group is an alternating group or a finite classical group, Proc. Lond. Math. Soc. (3) 47 (2) (1983) 337–362, https://doi.org/10.1112/plms/s3-47.2.337
[55] I.N. Ponomarenko, The isomorphism problem for classes of graphs, Dokl. Akad. Nauk SSSR 304 (3) (1989) 552–556.
[56] G. Ringel, Das geschlecht des vollständigen paaren graphen, Abh. Math. Semin. Univ. Hamb. 28 (3) (1965) 139–150, https://doi.org/10.1007/BF02993245
[57] F. Harary, Graph Theory, Addison-Wesley, 1991.
[58] I.N. Ponomarenko, Polynomial isomorphism algorithm for graphs which do not pinch to $K_{3,g}$, J. Sov. Math. 34 (1986) 1819–1831, https://doi.org/10.1007/BF01095641
[59] H.L. Bodlaender, Necessary edges in k-chordalisations of graphs, J. Comb. Optim. 7 (3) (2003) 283–290, https://doi.org/10.1023/A:1027320705349
[60] H. Leimer, Optimal decomposition by clique separators, Discret. Math. 113 (1–3) (1993) 99–123, https://doi.org/10.1016/0012-365X(93)90510-Z
[61] M. Elberfeld, P. Schweitzer, Canonizing graphs of bounded tree width in logspace, ACM Trans. Comput. Theory 9 (3) (2017) 12:1–12:29, https://doi.org/10.1145/3132720
[62] K. Kawarabayashi, Graph isomorphism for bounded genus graphs in linear time, CoRR arXiv:1511.02460, 2015.
[63] K. Kawarabayashi, B. Mohar, R. Nedela, P. Zeman, Automorphisms and isomorphisms of maps in linear time, ACM Trans. Algorithms 21 (1) (2025) 6:1–6:32, https://doi.org/10.1145/3686798
[64] M. Cygan, P. Komosa, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, S. Saurabh, M. Wahlström, Randomized contractions meet lean decompositions, ACM Trans. Algorithms 17 (1) (2021) 6:1–6:30, https://doi.org/10.1145/3426738
[65] S. Oum, P.D. Seymour, Approximating clique-width and branch-width, J. Comb. Theory, Ser. B 96 (4) (2006) 514–528, https://doi.org/10.1016/J.JCTB.2005.10.006
[66] S. Oum, Rank-width is less than or equal to branch-width, J. Graph Theory 57 (3) (2008) 239–244, https://doi.org/10.1002/JGT.20280
[67] M. Grohe, D. Neuen, Canonisation and definability for graphs of bounded rank width, ACM Trans. Comput. Log. 24 (1) (2023) 6:1–6:31, https://doi.org/10.1145/3568025
[68] P. Ohlmann, M. Pilipczuk, W. Przybyszewski, S. Torunczyk, Canonical decompositions in monadically stable and bounded shrubdepth graph classes, in: K. Etessami, U. Feige, G. Puppis (Eds.), 50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany, Vol. 261 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum Für Informatik, 2023, pp. 135:1–135:17, https://doi.org/10.4230/LIPICS.ICALP.2023.135
[69] M. DeVos, O. Kwon, S. Oum, Branch-depth: generalizing tree-depth of graphs, Eur. J. Comb. 90 (2020) 103186, https://doi.org/10.1016/J.EJC.2020.103186
[70] S. Oum, Rank-width and well-quasi-ordering, SIAM J. Discret. Math. 22 (2) (2008) 666–682, https://doi.org/10.1137/050629616
[71] V. Arvind, R. Nedela, I. Ponomarenko, P. Zeman, Testing isomorphism of chordal graphs of bounded leafage is fixed-parameter tractable (extended abstract), in: M.A. Bekos, M. Kaufmann (Eds.), Graph-Theoretic Concepts in Computer Science - 48th International Workshop, WG 2022, Tübingen, Germany, June 22-24, 2022, Revised Selected Papers, Vol. 13453 of Lecture Notes in Computer Science, Springer, 2022, pp. 29–42, https://doi.org/10.1007/978-3-031-15914-5_3
[72] D.A. Çagirici, P. Hlinený, Isomorphism testing for t-graphs in FPT, in: P. Mutzel, Md. S. Rahman, Slamin (Eds.), WALCOM: Algorithms and Computation - 16th International Conference and Workshops, WALCOM 2022, Jember, Indonesia, March 24-26, 2022, Proceedings, Vol. 13174 of Lecture Notes in Computer Science, Springer, 2022, pp. 239–250, https://doi.org/10.1007/978-3-030-96731-4_20
[73] É. Bonnet, E.J. Kim, S. Thomassé, R. Watrigant, Twin-width I: tractable FO model checking, J. ACM 69 (1) (2022) 3:1–3:46, https://doi.org/10.1145/3486655
[74] É. Bonnet, C. Geniet, E.J. Kim, S. Thomassé, R. Watrigant, Twin-width II: small classes, Comb. Theory 2 (2) (2022), https://doi.org/10.5070/c62257876
[75] É. Bonnet, C. Geniet, E.J. Kim, S. Thomassé, R. Watrigant, Twin-width III: max independent set, min dominating set, and coloring, SIAM J. Comput. 53 (5) (2024) 1602–1640, https://doi.org/10.1137/21m142188x
[76] É. Bonnet, U. Giocanti, P.O. de Mendez, P. Simon, S. Thomassé, S. Torunczyk, Twin-width IV: ordered graphs and matrices, J. ACM 71 (3) (2024) 21, https://doi.org/10.1145/3651151
[77] É. Bonnet, E.J. Kim, A. Reinald, S. Thomassé, R. Watrigant, Twin-width and polynomial kernels, Algorithmica 84 (11) (2022) 3300–3337, https://doi.org/10.1007/s00453-022-00965-5
[78] C. Geniet, S. Thomassé, First order logic and twin-width in tournaments and dense oriented graphs, Eur. J. Comb. 132 (2026) 104247, https://doi.org/10.1016/j.ejc.2025.104247
[79] P. Bergé, É. Bonnet, H. Déprés, Deciding twin-width at most 4 is np-complete, in: M. Bojanczyk, E. Merelli, D. P. Woodruff (Eds.), 49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France, Vol. 229 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum Für Informatik, 2022, pp. 18:1–18:20, https://doi.org/10.4230/LIPIcs.ICALP.2022.18
[80] M. Grohe, D. Neuen, Isomorphism for tournaments of small twin width, in: K. Bringmann, M. Grohe, G. Puppis, O. Svensson (Eds.), 51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia, Vol. 297 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum Für Informatik, 2024, pp. 78:1–78:20, https://doi.org/10.4230/LIPIcs.ICALP.2024.78
[81] J. Matousek, Lectures on Discrete Geometry, Vol. 212 of Graduate Texts in Mathematics, Springer, 2002, https://doi.org/10.1007/978-1-4613-0039-7