Canonizing Graphs of Bounded Rank-Width in Parallel via Weisfeiler–Leman

Michael Levet[1]
Department of Computer Science, College of Charleston, SC, USA

Puck Rombach
Department of Mathematics and Statistics, University of Vermont, Burlington, VT, USA

Nicholas Sieger
Department of Mathematics, University of California San Diego, La Jolla, CA, USA

Abstract

In this paper, we show that computing canonical labelings of graphs of bounded rank-width is in TC^2. Our approach builds on the framework of Köbler & Verbitsky (CSR 2008), who established the analogous result for graphs of bounded treewidth. Here, we use the framework of Grohe & Neuen (*ACM Trans. Comput. Log.*, 2023) to enumerate separators via *split-pairs* and *flip functions*. In order to control the depth of our circuit, we leverage the fact that any graph of rank-width k admits a rank decomposition of width $\leq 2k$ and height $O(\log n)$ (Courcelle & Kanté, WG 2007). This allows us to utilize an idea from Wagner (CSR 2011) of tracking the depth of the recursion in our computation.

Furthermore, after splitting the graph into connected components, it is necessary to decide isomorphism of said components in TC^1. To this end, we extend the work of Grohe & Neuen (*ibid.*) to show that the $(6k+3)$-dimensional Weisfeiler–Leman (WL) algorithm can identify graphs of rank-width k using only $O(\log n)$ rounds. As a consequence, we obtain that graphs of bounded rank-width are identified by $\mathsf{FO}+\mathsf{C}$ formulas with $6k+4$ variables and quantifier depth $O(\log n)$. Prior to this paper, isomorphism testing for graphs of bounded rank-width was not known to be in NC.

2012 ACM Subject Classification Theory of computation → Finite Model Theory; Theory of computation → Circuit complexity; Theory of computation → Graph algorithms analysis; Mathematics of computing → Graph algorithms

Keywords and phrases Graph Isomorphism, Weisfeiler–Leman, Rank-Width, Canonization, Descriptive Complexity, Circuit Complexity

Digital Object Identifier 10.4230/LIPIcs.SWAT.2024.32

Related Version *Full Version*: `https://arxiv.org/abs/2306.17777`

Funding This work was completed in part at the 2022 Graduate Research Workshop in Combinatorics, which was supported in part by NSF grant #1953985 and a generous award from the Combinatorics Foundation.
Michael Levet: Partially supported by J. A. Grochow's NSF award CISE-2047756 and the University of Colorado Boulder, Department of Computer Science Summer Research Fellowship.

1 Introduction

The Graph Isomorphism problem (GI) takes as input two graphs G and H, and asks if there exists an isomorphism $\varphi : V(G) \to V(H)$. GI is in particular conjectured to be NP-intermediate; that is, belonging to NP but neither in P nor NP-complete [36]. Algorithmically, the best known upper-bound is $n^{\Theta(\log^2 n)}$, due to Babai [3]. It remains open as to whether

[1] Corresponding author

GI belongs to P. There is considerable evidence suggesting that GI is not NP-complete [42, 6, 30, 3, 33, 1, 39]. In a precise sense, GI sits between linear and multilinear algebra. For any field $\mathbb{F}$, GI belongs to $\mathbb{F}$-Tensor Isomorphism ($\mathsf{TI}_{\mathbb{F}}$). When $\mathbb{F}$ is finite, $\mathsf{TI}_{\mathbb{F}} \subseteq \mathsf{NP} \cap \mathsf{coAM}$ [16, 17]. In contrast, the best known lower-bound for GI is DET [44], which contains NL and is a subclass of TC^1. It is thus natural to inquire as to families of graphs where isomorphism is decidable in sub-classes of DET.

There has been considerable work on efficient algorithms for special families of graphs. Sparse graphs, in particular, have received considerable attention from the perspective of polynomial-time computation, though less is known for dense graphs– see [24] for discussion. The story is similar in the setting of NC isomorphism tests, with considerable work on planar graphs (see the references in [12]) and graphs of bounded treewidth [25, 46, 11], culminating in L-completeness results for both families (see [12, 43] for planar graphs, and and [15] for graphs of bounded treewidth). Isomorphism testing for graphs of bounded genus is also L-complete [14]. We now turn to dense graphs. An NC isomorphism test is known for graphs of bounded eigenvalue multiplicity [2]. The isomorphism problems for interval graphs [32] and Helly ciruclar-arc graphs [35] are both L-complete.

The k-dimensional Weisfeiler–Leman algorithm (k-WL) serves as a key combinatorial tool in GI. It works by iteratively coloring k-tuples of vertices in an isomorphism-invariant manner. On its own, Weisfeiler–Leman serves as an efficient polynomial-time isomorphism test for several families of graphs, including planar graphs [31, 21], graphs of bounded genus [18, 20], and graphs for which a specified minor H is forbidden [19]. WL even serves as an NC isomorphism test for graphs of bounded treewidth [25] and planar graphs [25, 45, 21]. Despite the success of WL, it is insufficient to place GI into P [7, 40]. Nonetheless, WL remains an active area of research. For instance, Babai's quasipolynomial-time algorithm [3] combines $O(\log n)$-WL with group-theoretic techniques.

Graphs of bounded rank-width have only recently received attention from the perspective of isomorphism testing. Grohe & Schweitzer [24] gave the first polynomial-time isomorphism test for graphs of bounded rank-width. In particular, their isomorphism test ran in time $n^{f(k)}$, where $f(k)$ was a non-elementary function of the rank-width k. Subsequently, Grohe & Neuen [22] showed that graphs of rank-width k have Weisfeiler–Leman dimension $\leq 3k+5$, which yields an $O(n^{3k+6} \log n)$-time isomorphism test and also the first polynomial-time canonical labeling procedure for this family. In particular, it is open as to whether graphs of bounded rank-width admit NC or FPT isomorphism tests. This is in contrast to graphs of bounded treewidth, where NC [25, 34, 46, 11, 15] and FPT [38, 23] isomorphism tests are well-known.

Closely related to Graph Isomorphism is Graph Canonization, which for a class $\mathcal{C}$ of graphs, asks for a function $F : \mathcal{C} \to \mathcal{C}$ such that for all $X, Y \in \mathcal{C}$, $X \cong F(X)$ and $X \cong Y \iff F(X) = F(Y)$. Graph Isomorphism reduces to Graph Canonization, and the converse remains open. Nonetheless, efficient canonization procedures have often followed efficient isomorphism tests, usually with non-trivial work– see e.g., [29, 22, 34, 46, 15, 4].

Main Results. In this paper, we investigate the parallel and descriptive complexities of identifying and canonizing graphs of bounded rank-width, using the Weisfeiler–Leman algorithm.

▶ **Theorem 1.** *Let G be a graph on n vertices, of rank-width k. We can compute a canonical labeling for G using a TC circuit of depth $O(\log^2 n)$ and size $n^{O(16^k)}$.*

Our approach in proving Thm. 1 was inspired by the previous work of Köbler & Verbitsky [34], who established the analogous result for graphs of bounded treewidth. Köbler & Verbitsky crucially utilized the fact that graphs of treewidth k admit *balanced* separators of size $k+1$,

where removing such a separator leaves connected components each of size $\leq n/2$. This ensures that the height of their recursion tree is $O(\log n)$. For graphs of bounded rank-width, we are unable to identify such separators. Instead, we leverage the framework of Grohe & Neuen to descend along a rank decomposition, producing a canonical labeling along the way. To ensure that our choices are canonical, we utilize the Weisfeiler–Leman algorithm. As a first step, we will establish the following:

▶ **Theorem 2.** *The $(6k+3)$-dimensional Weisfeiler–Leman algorithm identifies graphs of rank-width k in $O(\log n)$ rounds.*

Combining Thm. 2 with the parallel WL implementation of Grohe & Verbitsky [25], we obtain the first NC bound for isomorphism testing of graphs of bounded rank-width. This is a crucial ingredient in obtaining the TC^2 bound for Thm. 1.

▶ **Corollary 3.** *Let G be a graph of rank-width $O(1)$, and let H be arbitrary. We can decide isomorphism between G and H in TC^1.*

Furthermore, in light of the close connections between Weisfeiler–Leman and $\mathsf{FO}+\mathsf{C}$ [29, 7], we obtain the following corollary. Let $\mathcal{C}_{m,r}$ denote the m-variable fragment of $\mathsf{FO}+\mathsf{C}$ where the formulas have quantifier depth at most r (see Sec. 2.3).

▶ **Corollary 4.** *For every graph G of rank-width at most k, there is a sentence $\varphi_G \in \mathcal{C}_{6k+4,O(\log n)}$ that characterizes G up to isomorphism. That is, whenever $H \ncong G$, we have that $G \models \varphi_G$ and $H \not\models \varphi_G$.*

We will discuss shortly the proof technique for Thm. 2. We first discuss how we will utilize Thm. 2 to establish Thm. 1. As $(6k+3)$-WL identifies all graphs of rank-width $\leq k$ in $O(\log n)$ rounds, $(10k+3)$-WL identifies the orbits of sequences of vertices of length $\leq 4k$ (see Lem. 21). By applying $(10k+3)$-WL for $O(\log n)$ rounds at each recursive call to our canonization procedure, we give canonical labelings to the various parallel choices considered by the algorithm. While there exists a suitable rank decomposition of height $O(\log n)$ [8], it is open whether such a decomposition can be computed in NC [10]. Instead of explicitly constructing a rank decomposition, we instead track the depth of our recursion tree. By leveraging the framework of Grohe & Neuen [22], we show that one of our parallel computations witnesses the balanced rank decomposition of Courcelle & Kanté [8].

We will now outline the proof strategy for Thm. 2. Our work follows closely the strategy of Grohe & Neuen [22]. We again combine the balanced rank decomposition from [8] with a careful analysis of the pebbling strategy of Grohe & Neuen [22]. In parts of their argument, Grohe & Neuen utilize (an analysis of) the stable coloring of 1-WL. For a graph G, Grohe & Neuen [22, Sec. 3] construct an auxiliary graph that they call the *flipped graph*, whose construction depends on a specified set of vertices called a *split pair* and a coloring of the vertices. While the flipped graph is compatible with any vertex coloring, Grohe & Neuen [22, Lem. 3.6] crucially utilize the *stable coloring* of 1-WL to show that WL can detect which edges are present in the flipped graph. Even though we allow for higher-dimensional WL, the restriction of $O(\log n)$ rounds creates a technical difficulty in adapting [22, Lem. 3.6].

To resolve this issue, we consider a *different* notion of flipped graph– namely, a vertex colored variant introduced in [22, Sec. 5]. In this second definition, edges of the flipped graph depend only on the split pair and not the coloring. Grohe & Neuen established [22, Lem. 5.6], which is analogous to their Lem. 3.6. The proof of their Lem. 5.6 depends only on the structure of the graph and *not* the vertex colorings. In particular, Weisfeiler–Leman can take advantage of [22, Lem. 5.6] within $O(\log n)$ iterations.

The second such place where Grohe & Neuen rely on the stable coloring of 1-WL to detect the connected components of the flipped graph. We will show that 2-WL can in fact identify these components in $O(\log n)$ rounds. We further reduce the round complexity via a simple observation. In the pebble game characterization, if Duplicator fails to respect connected components of the flipped graph, Spoiler can win in $O(\log n)$ rounds *without descending down the rank decomposition.* Otherwise, Spoiler only needs a constant number of rounds to descend to a child node in the rank decomposition. In either case, we only need $O(\log n)$ rounds total, which yields a TC^1 isomorphism test.

In the process of our work, we came across a result of Bodlaender [5], who showed that any graph of treewidth k admits a *binary* tree decomposition of width $\leq 3k+2$ and height $O(\log n)$. Using Bodlaender's result, we were able to modestly improve the descriptive complexity for graphs of bounded treewidth.

▶ **Theorem 5.** *The $(3k+6)$-dimensional Weisfeiler–Leman algorithm identifies graphs of treewidth k in $O(\log n)$ rounds.*

In light of the above theorem, we obtain the following improvement in the descriptive complexity for graphs of bounded treewidth.

▶ **Corollary 6.** *Let G be a graph of treewidth k. Then there exists a formula $\varphi_G \in \mathcal{C}_{3k+7,O(\log n)}$ that identifies G up to isomorphism. That is, for any $H \ncong G$, $G \models \varphi_G$ and $H \not\models \varphi_G$.*

2 Preliminaries

2.1 Weisfeiler–Leman

We begin by recalling the Weisfeiler–Leman algorithm for graphs, which computes an isomorphism-invariant coloring. Let G be a graph on n vertices, let $\chi : V(G) \to [n]$ be a coloring of the vertices, and let $k \geq 2$ be an integer. The k-dimensional Weisfeiler–Leman, or k-WL, algorithm begins by constructing an initial coloring $\chi_0 : V(G)^k \to \mathcal{K}$, where $\mathcal{K}$ is our set of colors, by assigning each k-tuple a color based on its isomorphism type under the coloring χ.[2] Two k-tuples $(v_1, \ldots, v_k) \in V(G)^k$ and $(u_1, \ldots, u_k) \in V(G)^k$ receive the same color under χ_0 if and only if the following conditions all hold

- For all i, j, $v_i = v_j \Leftrightarrow u_i = u_j$.
- The map $v_i \mapsto u_i$ (for all $i \in [k]$) is an isomorphism of the induced subgraphs $G[\{v_1, \ldots, v_k\}]$ and $G[\{u_1, \ldots, u_k\}]$
- $\chi(u_i) = \chi(v_i)$ for all $i \in [k]$.

For $r \geq 0$, the coloring computed at the rth iteration of Weisfeiler–Leman is refined as follows. For a k-tuple $\overline{v} = (v_1, \ldots, v_k)$ and a vertex $x \in V(G)$, define

$$\overline{v}(v_i/x) = (v_1, \ldots, v_{i-1}, x, v_{i+1}, \ldots, v_k).$$

The coloring computed at the $(r+1)$st iteration, denoted χ_{r+1}, stores the color of the given k-tuple $\overline{v}$ at the rth iteration, as well as the colors under χ_r of the k-tuples obtained by substituting a single vertex in $\overline{v}$ for another vertex x. We examine this multiset of colors over all such vertices x. This is formalized as follows:

$$\chi_{r+1}(\overline{v}) = (\chi_r(\overline{v}), \{\!\{(\chi_r(\overline{v}(v_1/x)), \ldots, \chi_r(\overline{v}(v_k/x)) \,|\, x \in V(G)\}\!\}),$$

[2] Note that for k-WL applied to two graphs G and H, each of order n, there are at most $2n^k$ color classes. So without loss of generality, we may take $\mathcal{K} = [2n^k]$.

where $\{\!\{\cdot\}\!\}$ denotes a multiset. Note that the coloring χ_r computed at iteration r induces a partition of $V(G)^k$ into color classes. The Weisfeiler–Leman algorithm terminates when this partition is not refined, that is, when the partition induced by χ_{r+1} is identical to that induced by χ_r. The final coloring is referred to as the *stable coloring*, which we denote $\chi_\infty := \chi_r$.

The 1-dimensional Weisfeiler–Leman algorithm, sometimes referred to as *Color Refinement*, works nearly identically. The initial coloring is that provided by the vertex coloring for the input graph. For the refinement step, we have that: $\chi_{r+1}(u) = (\chi_r(u), \{\!\{\chi_r(v) : v \in N(u)\}\!\})$. We have that 1-WL terminates when the partition on the vertices is not refined.

As we are interested in both the Weisfeiler–Leman dimension and the number of rounds, we will use the following notation.

▶ **Definition 7.** *Let $k \geq 1$ and $r \geq 1$ be integers. The (k, r)-WL algorithm is obtained by running k-WL for r rounds. Here, the initial coloring counts as the first round.*

Let S be a sequence of vertices. The *individualize-and-refine* paradigm works first by assigning each vertex in S a unique color. We then run (k, r)-WL starting from this choice of initial coloring. We denote the coloring computed by (k, r)-WL after individualizing S as $\chi^S_{k,r}$. When there is ambiguity about the graph G in question, we will for clarity write $\chi^{S,G}_{k,r}$.

For two graphs G and H, we say that (k, r)-WL *distinguishes* G and H if there is some color c such that: $|\{v \in V(G)^k : \chi_{G,k,r}(v) = c\}| \neq |\{w \in V(H)^k : \chi_{H,k,r}(w) = c\}|$. Additionally, (k, r)-WL *identifies* a graph G if (k, r)-WL distinguishes G from every graph H such that $G \ncong H$.

▶ Remark 8. Grohe & Verbitsky [25] previously showed that for fixed k, the classical k-dimensional Weisfeiler–Leman algorithm for graphs can be effectively parallelized. Precisely, each iteration (including the initial coloring) can be implemented using a logspace uniform TC^0 circuit.

2.2 Pebbling Game

We recall the bijective pebble game introduced by [26, 27] for WL on graphs. This game is often used to show that two graphs X and Y cannot be distinguished by k-WL. The game is an Ehrenfeucht–Fraïssé game (c.f., [13, 37]), with two players: Spoiler and Duplicator. We begin with $k + 1$ pairs of pebbles. Prior to the start of the game, each pebble pair (p_i, p'_i) is initially placed either beside the graphs or on a given pair of vertices $v_i \mapsto v'_i$ (where $v_i \in V(X), v'_i \in V(Y)$). We refer to this initial configuration for X as $\overline{v}$, and this initial configuration for Y as $\overline{v'}$. Each round r proceeds as follows.

1. Spoiler picks up a pair of pebbles (p_i, p'_i).
2. Duplicator chooses a bijection $f_r : V(X) \to V(Y)$ (we emphasize that the bijection chosen depends on the round and, implicitly, the pebbling configuration at the start of said round).
3. Spoiler places p_i on some vertex $v \in V(X)$. Then p'_i is placed on $f(v)$.

Let $v_1, \ldots, v_m$ be the vertices of X pebbled at the end of round r of the game, and let $v'_1, \ldots, v'_m$ be the corresponding pebbled vertices of Y. Spoiler wins precisely if the map $v_\ell \mapsto v'_\ell$ is not an isomorphism of the induced subgraphs $X[\{v_1, \ldots, v_m\}]$ and $Y[\{v'_1, \ldots, v'_m\}]$. Duplicator wins otherwise. Spoiler wins, by definition, at round 0 if X and Y do not have the same number of vertices. We note that $\overline{v}$ and $\overline{v'}$ are not distinguished by the first r rounds of k-WL if and only if Duplicator wins the first r rounds of the $(k + 1)$-pebble game [26, 27, 7].

We establish a helper lemma, which effectively states that Duplicator must respect connected components of pebbled vertices.

▶ **Lemma 9.** *Let G, H be graphs on n vertices. Suppose that $(u, v) \mapsto (u', v')$ have been pebbled. Furthermore, suppose that u, v belong to the same connected component of G, while u', v' belong to different connected components of H. Then Spoiler can win using 1 additional pebble and $O(\log n)$ rounds.*

2.3 Logics

We recall key notions of first-order logic. We have a countable set of variables $\{x_1, x_2, \ldots\}$. Formulas are defined inductively. For the basis, we have that $x_i = x_j$ is a formula for all pairs of variables. Now if φ_1, φ_2 are formulas, then so are the following: $\varphi_1 \wedge \varphi_2, \varphi_1 \vee \varphi_2, \neg\varphi_1, \exists x_i\, \varphi_1$, and $\forall x_i\, \varphi_1$. In order to define logics on graphs, we add a relation $E(x, y)$, where $E(x, y) = 1$ if and only if $\{x, y\}$ is an edge of our graph, and 0 otherwise. In keeping with the conventions of [7], we refer to the first-order logic with relation E as $\mathcal{L}$ and its k-variable fragment as $\mathcal{L}_k$. We refer to the logic $\mathcal{C}$ as the logic obtained by adding counting quantifiers $\exists^{\geq n} x\, \varphi$ (there exist at least n elements x that satisfy φ) and $\exists! n\, x\, \varphi$ (there exist exactly n elements x that satisfy φ) and its k-variable fragment as $\mathcal{C}_k$.

The *quantifier depth* of a formula φ (belonging to either $\mathcal{L}$ or $\mathcal{C}$) is the depth of its quantifier nesting. We denote the quantifier depth of φ as $\mathrm{qd}(\varphi)$ This is defined inductively as follows.

- If φ is atomic, then $\mathrm{qd}(\varphi) = 0$.
- $\mathrm{qd}(\neg\varphi) = \mathrm{qd}(\varphi)$.
- $\mathrm{qd}(\varphi_1 \vee \varphi_2) = \mathrm{qd}(\varphi_1 \wedge \varphi_2) = \max\{\mathrm{qd}(\varphi_1), \mathrm{qd}(\varphi_2)\}$.
- $\mathrm{qd}(Qx\, \varphi) = \mathrm{qd}(\varphi) + 1$, where Q is a quantifier in the logic.

We denote the fragment of $\mathcal{L}_k$ (respectively, $\mathcal{C}_k$) where the formulas have quantifier depth at most r as $\mathcal{L}_{k,r}$ (respectively, $\mathcal{C}_{k,r}$). Let $\overline{v} \in V(X)^k, \overline{v'} \in V(Y)^k$. We note that $\overline{v}, \overline{v'}$ are distinguished by (k, r)-WL if and only if there exists a formula $\varphi \in \mathcal{C}_{k+1,r}$ such that $(X, \overline{v}) \models \varphi$ and $(Y, \overline{v'}) \not\models \varphi$ [29, 7].

2.4 Rank-Width

Oum & Seymour [28] introduced the rank-width parameter to measure the width of a certain hierarchical decomposition of graphs. The goal is to intuitively split the vertices of a graph along cuts of low complexity in a hierarchical fashion. Here, the complexity is the $\mathbb{F}_2$-rank of the matrix capturing the adjacencies crossing the cut.

Precisely, let G be a graph, and let $X, Y \subseteq V(G)$. Define $M(X, Y) \in \mathbb{F}_2^{X \times Y}$ to be the matrix where $(M(X, Y))_{uv} = 1$ if and only if $uv \in E(G)$. That is, $M(X, Y)$ is the submatrix of the adjacency matrix whose rows are indexed by X and whose columns are indexed by Y. Denote $\rho(X) := \mathrm{rk}_{\mathbb{F}_2}(M(X, \overline{X}))$.

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

- For the root r of T, $\gamma(r) = V(G)$,
- For an internal node $t \in V(T)$, denote the children of t as s_1, s_2. For every internal node t, we have that $\gamma(t) = \gamma(s_1) \cup \gamma(s_2)$, and $\gamma(s_1) \cap \gamma(s_2) = \emptyset$.
- For any leaf $t \in V(T)$, $|\gamma(t)| = 1$.

▶ Remark 10. Let $L(T)$ be the set of leaves of T. Instead of providing γ, we can equivalently define a bijection $f : V(G) \to L(T)$. By the second condition of a rank decomposition, f completely determines γ.

The *width* of a rank decomposition (T, γ) is: $\mathrm{wd}(T, \gamma) := \max\{\rho_G(\gamma(t)) : t \in V(T)\}$. The *rank-width* of a graph G is: $\mathrm{rw}(G) := \min\{\mathrm{wd}(T, \gamma) : (T, \gamma) \text{ is a rank decomposition of } G\}$.

The parameter rank-width is closely related to the parameter clique width, introduced by Courcelle & Olariu [9]. Oum & Seymour [28] showed that: $\mathrm{rw}(G) \leq \mathrm{cw}(G) \leq 2^{\mathrm{rw}(G)+1} - 1$. Denote $\mathrm{tw}(G)$ to be the treewidth of G. Oum [41] showed that $\mathrm{rw}(G) \leq \mathrm{tw}(G) + 1$. Note that $\mathrm{tw}(G)$ cannot be bounded in terms of $\mathrm{rw}(G)$; for instance, the complete graph K_n has $\mathrm{rw}(K_n) = 1$ but $\mathrm{tw}(K_n) = n - 1$.

3 Weisfeiler–Leman for Graphs of Bounded Rank-Width

3.1 Split Pairs and Flip Functions

In designing a pebbling strategy for graphs of bounded rank-width, Grohe & Neuen [22] sought to pebble a set of vertices $X \subseteq V(G)$ such that $\rho(X) \leq k$ and pebbling X partitions the remaining vertices into sets $C_1, \dots, C_\ell$ that can be treated independently. Furthermore, we want for each $i \in [\ell]$ that either $C_i \subseteq X$ or $C_i \subseteq \overline{X}$. As there can be many edges between X and $\overline{X}$, this is hard to accomplish in general. To this end, Grohe & Neuen [22] utilized split pairs and flip functions. We will now recall their framework.

Let $G(V, E, \chi)$ be a colored graph on n vertices, and suppose the rank-width of G is at most k. Let $X \subseteq V(G)$. For $v \in X$, define $\mathrm{vec}_X(v) = (a_{v,w})_{w \in \overline{X}} \in \mathbb{F}_2^{\overline{X}}$, where $a_{v,w} = 1$ if and only if $vw \in E(G)$. For $S \subseteq X$, define $\mathrm{vec}_X(S) = \{\mathrm{vec}_X(v) : v \in S\}$. A *split pair* for X is a pair (A, B) such that:

(a) $A \subseteq X$, and $B \subseteq \overline{X}$,
(b) $\mathrm{vec}_X(A)$ forms a linear basis for $\langle \mathrm{vec}_X(X) \rangle$, and
(c) $\mathrm{vec}_{\overline{X}}(B)$ forms a linear basis for $\langle \mathrm{vec}_{\overline{X}}(\overline{X}) \rangle$.

An *ordered split pair* for X is a pair $((a_1, \dots, a_q), (b_1, \dots, b_p))$ such that $(\{a_1, \dots, a_q\}, \{b_1, \dots, b_p\})$ is a split pair for X.

Let $G(V, E, \chi)$ be a colored graph on n vertices, and suppose the rank-width of G is at most k. An *ordered split pair*, $(\overline{a}, \overline{b})$, of order at most $2k$ is a pair $(\overline{a}, \overline{b})$, where $\overline{a}, \overline{b} \in V(G)^{\leq 2k}$. For $v, w \in V(G)$, we say that $v \approx_{(\overline{a},\overline{b})} w$ if $N(v) \cap (\overline{a}, \overline{b}) = N(w) \cap (\overline{a}, \overline{b})$ (here, we consider $N(v) \cap (\overline{a}, \overline{b})$ as a set). Observe that $\approx_{(\overline{a},\overline{b})}$ forms an equivalence relation. For $(\overline{a}, \overline{b}) \in V(G)^{\leq 2k}$, let $2^{\overline{a} \cup \overline{b}}$ be the set of all subsets of $\overline{a} \cup \overline{b} \subseteq V(G)$, where we abuse notation by considering $\overline{a}, \overline{b}$ as subsets of $V(G)$. A *flip extension* of an ordered split pair $(\overline{a}, \overline{b})$ is a tuple:

$$\overline{s} := \left(\overline{a}, \overline{b}, f : \left(2^{\overline{a} \cup \overline{b}}\right)^2 \to [n] \cup \{\bot\}\right),$$

such that for all $M, N \in 2^{\overline{a} \cup \overline{b}}$ with $M \neq N$, either $f(M, N) = \bot$ or $f(N, M) = \bot$. There is no restriction on $f(M, N)$ if $M = N$. For $v, w \in V(G)$, we say that $v \approx_{\overline{s}} w$ if $v \approx_{(\overline{a},\overline{b})} w$. Denote $[v]_{\approx \overline{s}}$ to be the equivalence class of v with respect to $\approx_{\overline{s}}$. Define the *flipped graph* $G^{\overline{s}} = (V, E^{\overline{s}}, \chi, \overline{a}, \overline{b})$, where $V(G^{\overline{s}}) = V(G)$,

$$\begin{aligned} E^{\overline{s}} := &\{vw \in E(G) : f(N(v) \cap (\overline{a}, \overline{b}), N(w) \cap (\overline{a}, \overline{b})) = d \in [n] \wedge |N(v) \cap [w]_{\approx \overline{s}}| < d\} \\ &\cup \{vw \notin E(G) : f(N(v) \cap (\overline{a}, \overline{b}), N(w) \cap (\overline{a}, \overline{b})) = d \in [n] \wedge |N(v) \cap [w]_{\approx \overline{s}}| \geq d\}, \end{aligned}$$

and χ is the same coloring as in G. Denote $\mathrm{Comp}(G, \overline{s}) \subseteq 2^{V(G)}$ be the set of vertex sets of the connected components of $G^{\overline{s}}$. Observe that $\mathrm{Comp}(G, \overline{s})$ forms a partition of $V(G)$. Grohe & Neuen [22] established that for any choice $(\overline{a}, \overline{b})$ of split pair, there exists a suitable flip function; and thus, a suitable flip extension.

▶ **Lemma 11** ([22, Lem. 5.6])**.** *Let G be a (colored) graph, and let $X \subseteq V(G)$. Furthermore, let $(\overline{a}, \overline{b})$ be an ordered split pair for X. Then there exists a flip extension $\overline{s} := (\overline{a}, \overline{b}, f)$ such that $C \subseteq X$ or $C \subseteq \overline{X}$ for every $C \in Comp(G, \overline{s})$.*

Grohe & Neuen [22, Sec. 5] considered uncolored flipped graphs. As the conditions for determining the edges of the flipped graph do not depend on the vertex colors, [22, Lem. 5.6] holds in our setting.

We now turn to showing that the flip extensions preserve both isomorphism and the effects of Weisfeiler–Leman. To do so, we consider vertex colorings χ that refine the coloring $\chi_{1,3}$ computed by $(1,3)$-WL. The advantage of incorporating such a coloring on the vertices, is that it encodes some data about how the vertices of G interact with the specified split pair. Furthermore, the colorings computed by Weisfeiler–Leman are invariant under isomorphism. We take advantage of this to establish that the flipped graph preserves both the isomorphism problem (Lem. 12) and the effects of Weisfeiler–Leman (Lem. 13). For a graph G of rank-width k, we will be running $(6k+3, O(\log n))$, and so we may assume without loss of generality that the vertices of G have been colored according to $(1,3)$-WL.

▶ **Lemma 12.** *Let G, H be graphs, and let $\overline{s} = (\overline{a}, \overline{b}, f), \overline{s'} = (\overline{a'}, \overline{b'}, f)$ be flip extensions for G, H, respectively (we stress that the function f appearing in $\overline{s}$ is the same as that appearing in $\overline{s'}$). Let $k \geq 1, r \geq 3$. Consider the colorings $\chi_{k,r}^{(\overline{a},\overline{b}),G}, \chi_{k,r}^{(\overline{a'},\overline{b'}),H}$ obtained by individualizing $(\overline{a}, \overline{b}) \mapsto (\overline{a'}, \overline{b'})$ and applying (k,r)-WL.*

Let $\varphi : V(G) \to V(H)$ be a bijection. We have that φ is an isomorphism of the colored graphs $(G, \chi_{k,r}^{(\overline{a},\overline{b}),G}) \cong (H, \chi_{k,r}^{(\overline{a'},\overline{b'}),H})$ if and only if φ is an isomorphism of $G^{\overline{s}} \cong H^{\overline{s'}}$.

▶ **Lemma 13** (cf. [22, Lem. 3.10])**.** *Let $G(V, E, \chi), G'(V', E', \chi')$ be colored graphs, and let $\overline{s} = (\overline{a}, \overline{b}, f)$ and $\overline{s'} = (\overline{a'}, \overline{b'}, f)$ be flip extensions (we are using the same flip function f for both $\overline{s}, \overline{s'}$). Let $\chi_{1,3}$ be the coloring resulting from individualizing $(\overline{a}, \overline{b}) \mapsto (\overline{a'}, \overline{b'})$ and running $(1,3)$-WL. Suppose that χ, χ' both refine $\chi_{1,3}$. Let $((\overline{v}, \overline{w})) = ((v_1, \ldots, v_\ell), (w_1, \ldots, w_\ell))$ be a position in the ℓ-pebble bijective pebble game. We have that Spoiler wins from $((\overline{v}, \overline{w}))$ in the ℓ-pebble, r-round game on (G, G') if and only if Spoiler wins from from $((\overline{v}, \overline{w}))$ in the ℓ-pebble, r-round game on $(G^{\overline{s}}, (G')^{\overline{s}})$.*

▶ **Corollary 14** (Compare rounds cf. [22, Corollary 3.12])**.** *Let $G(V, E, \chi), G'(V', E', \chi')$ be colored graphs, and let $\overline{s} = (\overline{a}, \overline{b}, f)$ and $\overline{s'} = (\overline{a'}, \overline{b'}, f)$ be flip extensions (we are using the same flip function f for both $\overline{s}, \overline{s'}$). Let $\chi_{1,3}$ be the coloring resulting from individualizing $(\overline{a}, \overline{b}) \mapsto (\overline{a'}, \overline{b'})$ and running $(1,3)$-WL. Suppose that χ, χ' both refine $\chi_{1,3}$.*

Let $\overline{v} \in V^k, \overline{v'} \in (V')^k$. Let C be a connected component of $G^{\overline{s}}$ such that $\chi(u) \neq \chi(w)$ for all $u \in C$ and all $w \in V \setminus C$. Let C' be a connected component of $(G')^{\overline{s'}}$ such that $\chi'(u') \neq \chi'(w')$ for all $u' \in C'$ and $w' \in V' \setminus C'$. Let $r \geq 1$. Suppose that: $(G[C], \chi_{1,r}^{\overline{v},G}) \ncong (G'[C'], \chi_{1,r}^{\overline{v'},G'})$. Let $\overline{w} := C \cap \overline{v}$ and $\overline{w'} := C' \cap \overline{v'}$. Then either: $(G[C], \chi_{1,r}^{\overline{w},G}) \ncong (G'[C'], \chi_{1,r}^{\overline{w'},G'})$, or r rounds of Color Refinement distinguishes $(G, \chi^{\overline{v}})$ from $(G', (\chi')^{\overline{v'}})$.

3.2 WL for Graphs of Bounded Rank-Width

Our goal in this section is to establish the following.

▶ **Theorem 15.** *Let G be a graph on n vertices of rank-width k, and let H be an arbitrary graph such that $G \not\cong H$. We have that the $(6k+3, O(\log n))$-WL algorithm will distinguish G from H.*

▶ **Definition 16** ([22, Definition 4.1])**.** *Let G be a graph, and let $X, X_1, X_2 \subseteq V(G)$ such that $X = X_1 \sqcup X_2$. Let (A, B) be a split pair for X, and let (A_i, B_i) $(i = 1, 2)$ be a split pair for X_i. We say that (A_i, B_i) are nice with respect to (A, B) if the following conditions hold:*
(a) *$A \cap X_i \subseteq A_i$ for each $i \in \{1, 2\}$, and*
(b) *$B_2 \cap X_1 \subseteq A_1$ and similarly $B_1 \cap X_2 \subseteq A_2$.*
A triple $((A, B), (A_1, B_1), (A_2, B_2))$ of ordered split pairs is nice if the underlying triple of unordered split pairs is nice.

▶ **Lemma 17** ([22, Lem. 4.2])**.** [3] *Let G be a graph, and let $X, X_1, X_2 \subseteq V(G)$ such that $X = X_1 \sqcup X_2$. Let (A, B) be a split pair for X. There exist nice split pairs (A_i, B_i) for X_i $(i = 1, 2)$ such that additionally $B_i \cap \overline{X_i} \subseteq B$.*

▶ **Definition 18.** *Let G be a graph. A component partition of G is a partition $\mathcal{P}$ of $V(G)$ such that every connected component appears in exactly one block of $\mathcal{P}$. That is, for every connected component C of G, there exists a $P \in \mathcal{P}$ such that $C \subseteq P$.*

▶ **Lemma 19** ([22, Observation 4.3])**.** *Let G, H be two non-isomorphic graphs, and let $\mathcal{P}, \mathcal{Q}$ be component partitions of G, H respectively. Let $\sigma : V(G) \to V(H)$ be a bijection. There exists a vertex v of G such that $G[P] \not\cong H[Q]$, where $P \in \mathcal{P}$ is the unique set containing v and $Q \in \mathcal{Q}$ is the unique set containing $\sigma(v)$.*

We now prove Thm. 15.

Proof Idea of Thm. 15 . We follow the strategy of [22, Thm. 4.4]. We will briefly discuss the how we modified the proof from [22]; the full proof will appear in the full version. Let $G(V, E, \chi_G)$ be a colored graph of rank width $\leq k$, and let H be an arbitrary graph such that $G \not\cong H$. By [8, Thm. 5], G admits a rank decomposition (T, γ) of width at most $2k$ where T has height at most $3 \cdot (\log(n) + 1)$.

We will show that Spoiler has a winning strategy in the $6k+3$ pebble game in $O(\log n)$ rounds. In a similar manner as in the proof of [22, Thm. 4.4], we will first argue that $12k+5$ pebbles suffice, and then show how to improve the bound to use only $6k+3$ pebbles.

Spoiler's strategy is to play along the rank decomposition (T, γ) starting from the root. As Spoiler proceeds down the tree, the non-isomorphism is confined to increasingly smaller parts of G and H. At a node $t \in V(T)$, Spoiler pebbles a split pair $(\overline{a}, \overline{b}) \mapsto (\overline{a'}, \overline{b'})$, where $(\overline{a}, \overline{b})$ corresponds to a flip extension $\overline{s} = (\overline{a}, \overline{b}, f)$ of $X = \gamma(t)$. Let $\overline{s'} := (\overline{a'}, \overline{b'}, f)$. Now to confine the non-isomorphism, Spoiler identifies, after individualizing the split pair and performing three steps of Color Refinement- the initial coloring and two refinement steps, a pair of non-isomorphic components $C \subseteq X, C' \subseteq V(H)$ in the flipped graphs $G^{\overline{s}}$ and $H^{\overline{s'}}$. In particular, Spoiler seeks to find such components C and C' such that C is increasingly

[3] Grohe & Neuen use in the proof of [22, Thm. 5.5] that [22, Lem. 4.2] holds for the flipped graphs they define in Section 5, and not just the earlier notion of flipped graphs they consider in Section 3. Hence, [22, Lem. 4.2] holds in our setting as well.

further from the root of T. Once Spoiler reaches a leaf node of T, Spoiler can quickly win. Spoiler places a pebble on a vertex in C and its image in C', under Duplicator's bijection at the given round.

We note that the three rounds of Color Refinement suffice for WL to detect the partitioning induced by the flip function, though it is not sufficiently powerful to detect the connected components of $G^{\overline{s}}$ and $H^{\overline{s'}}$. In the argument below, we will technically consider graphs where the refinement step uses $(2, O(\log n))$-WL. This ensures that after individualizing a vertex on a given component C, that the vertices of C receive different colors than those of $V(G) \setminus C$. This will eventually happen, and so in the pebble game characterization, we can continue to descend along T as if the vertices of C have been distinguished from $V(G) \setminus C$. This is a key point where our strategy deviates from that of [22, Thm. 4.4]. The remaining details will appear in the full version. ◀

4 Canonical Forms in Parallel

In this section, we will establish the following.

▶ **Theorem 20.** *Let G be a graph on n vertices, of rank-width k. We can compute a canonical labeling for G using a TC circuit of depth $O(\log^2 n)$ and size $n^{O(16^k)}$.*

We will prove Thm. 20 via the individualization-and-refinement paradigm. Our strategy is similar to that of Köbler & Verbitsky [34], who established the analogous result for treewidth. We will begin by briefly recalling their approach. Köbler & Verbitsky began by enumerating ordered sequences of vertices of length $\leq k+1$, testing whether each such sequence disconnected the graph. In particular, Köbler & Verbitsky crucially used the fact that a graph of treewidth k admits a so-called *balanced* separator S of size $\leq k+1$, which splits G into connected components each of size $\leq n/2$. Köbler & Verbitsky then colored the vertices of each connected component of $G - S$ according to how they connected back to S. As graphs of bounded treewidth are hereditary (closed under taking induced subgraphs), Köbler & Verbitsky were then able to recurse on the connected components. The existence of balanced separators guarantees that only $O(\log n)$ such recursive calls are needed.

Instead of relying on balanced separators, it is sufficient to guarantee that after $O(\log n)$ recursive calls, each connected component will be a singleton. To this end, we again leverage the result of [8], who showed that a graph of rank-width k admits a rank decomposition (T, γ) of width $\leq 2k$ and height $O(\log n)$.

Thus, we would intuitively like to descend along such a rank decomposition (T, γ) of width $\leq 2k$ and height $O(\log n)$. Fix a node $t \in V(T)$, and let t_1 be the left child and t_2 be the right child of t. We would then enumerate over all pairs of flip extensions $((\overline{a_1}, \overline{b_1}, f_1), (\overline{a_2}, \overline{b_2}, f_2))$, where intuitively $\overline{s_i} := (\overline{a_i}, \overline{b_i}, f_i)$ is a flip extension for $\gamma(t_i)$. Then for each $i = 1, 2$ and each component $C_i \in \mathrm{Comp}(G[\gamma(t)], \overline{s_i})$, we apply the construction recursively. Note that we are not able to efficiently compute a rank decomposition of width $\leq 2k$ and height $O(\log n)$. Nonetheless, Lem. 11 guarantees the existence of flip extensions that witness the decomposition of a fixed rank decomposition (T, γ). Following an idea of Wagner [46], we consider all possible flip extensions in parallel, and thus ensure that the flip extension which respects a fixed rank decomposition is considered by the algorithm. As we will show in Lem. 24, the existence of a rank decomposition of height $O(\log n)$ allows us to guarantee that at least one of the flip extensions considered by the algorithm will produce a labeling, and Lem. 25 will then guarantee that the minimum such labeling (which is the labeling the algorithm will return) is in fact canonical. Now to the details.

We first show that we can enumerate the split pairs in a canonical manner. To this end, we will need the following lemma, which is essentially well-known amongst those working on the Weisfeiler–Leman algorithm (cf., [29, 22]).

Let G be a graph. The (k,r)-Weisfeiler-Leman algorithm *determines orbits of ℓ-tuples* if, for every graph H, every $v \in V(G)^\ell$ and every $w \in V(H)^\ell$ such that $\chi_{k,r}(v) = \chi_{k,r}(w)$, there is an isomorphism $\varphi : V(G) \to V(H)$ such that $\varphi(v) = w$.

▶ **Lemma 21.** *Let $\mathcal{C}$ be a class of graphs such that (k,r)-WL identifies all (colored) $G \in \mathcal{C}$. Then for any $\ell \geq 1$ and all (colored) $G \in \mathcal{C}$, $(k+\ell,r)$-WL determines the orbits of all ℓ-tuples of vertices in G.*

By Thm. 2, we have that $(6k+3, O(\log n))$-WL identifies all graphs of rank-width k. As we will need to enumerate split pairs, which have length $\leq 4k$, we will run $(10k+3, O(\log n))$-WL at each stage. Lem. 21 ensures that enumerating the split pairs in color class order is canonical. Note that a flip function is represented as a tuple in $\{0,\dots,n\}^{2^{4k}}$. So for a fixed split pair $(\overline{a},\overline{b})$, we can canonically enumerate the flip functions in lexicographic order. Thus, flip extensions can be enumerated in a canonical order.

▶ Remark 22. Now let $(\overline{a},\overline{b})$ be a split pair on G and $(\overline{c},\overline{d})$ be a split pair on H such that $\chi_{10k+3,O(\log n)}((\overline{a},\overline{b})) = \chi_{10k+3,O(\log n)}((\overline{c},\overline{d}))$. Let f be a given flip function, and let $\overline{s} = (\overline{a},\overline{b},f), \overline{s'} = (\overline{c},\overline{d},f)$ be flip extensions. By Lem. 21, there is an isomorphism mapping $(\overline{a},\overline{b}) \mapsto (\overline{c},\overline{d})$. Hence, Lem. 12 provides that the flipped graphs $G^{\overline{s}}, H^{\overline{s'}}$ are isomorphic whenever $G \cong H$. In particular, if there is an isomorphism $\varphi : G \cong H$ mapping $(\overline{a},\overline{b}) \mapsto (\overline{c},\overline{d})$, then φ is also an isomorphism of $G^{\overline{s}} \cong H^{\overline{s'}}$.

▶ **Lemma 23.** *Let G be a graph, $X \subseteq V(G)$, and $\overline{s} = (\overline{a},\overline{b},f)$ be a flip extension for $G[X]$. We may write down the flipped graph $G^{\overline{s}}$ and identify the connected components of $G[X]^{\overline{s}}$ in L.*

We will now pause to outline the procedure for the reader. Let $\overline{s} := (\overline{a},\overline{b},f)$ be a flip extension for $V(G)$. We will first individualize $(\overline{a},\overline{b})$ and apply $(10k+3, O(\log n))$-WL to G. For each component $C \in \mathrm{Comp}(G,\overline{s})$, this will encode the isomorphism class of $G[C]$ (as $(6k+3, O(\log n))$-WL identifies all graphs of rank-width $\leq k$– see Thm. 15), as well as how $G[C]$ connects back to the rest of G. It is easy to see that for any two vertices v, w, if v, w receive the same color under $\chi^{(\overline{a},\overline{b})}_{10k+3,O(\log n)}$, then the following conditions hold:

(a) $N(v) \cap (\overline{a} \cup \overline{b}) = N(w) \cap (\overline{a} \cup \overline{b})$, and
(b) For any vertex u, $|N(v) \cap [u]_{\approx\overline{s}}| = |N(w) \cap [u]_{\approx\overline{s}}|$.

Intuitively, this coloring encodes how each given vertex connects to the rest of G. Precisely, let $G \cong H$ be graphs of rank-width $\leq k$, and suppose that the algorithm returns the labeling $\lambda : V(G) \to [n]$ for G and labeling $\kappa : V(H) \to [n]$ for H (where $n = |G| = |H|$). If $v, w \in V(G)$ belong to different components of $\mathrm{Comp}(G,\overline{s})$, then we need to ensure that $\{v,w\} \in E(G)$ if and only if $\{(\kappa^{-1} \circ \lambda)(v), (\kappa^{-1} \circ \lambda)(w)\} \in E(H)$. By the definition of the flipped graph (Sec. 3), conditions (a) and (b) determine precisely whether $\{v,w\} \in E(G)$.

By Lem. 23, we may write down the connected components for the flipped graph $G^{\overline{s}}$ in L. We will then sort these connected components in lexicographic order by color class, which is L-computable. It may be the case that for two connected components $C_i, C_j \in \mathrm{Comp}(G,\overline{s})$, $G[C_i]$ and $G[C_j]$ are isomorphic and connect to the rest of G in the same way, and so receive the same multiset of colors. In this case, we may arbitrarily choose whether $G[C_i]$ will be sorted before $G[C_j]$. The output will not depend on this particular choice, as there is an automorphism of G which exchanges the two components. Now for each $C \in \mathrm{Comp}(G,\overline{s})$, we will apply the procedure recursively on $G[C]$, incrementing the local depth variable by 1. If for each connected component of $\mathrm{Comp}(G,\overline{s})$ we are given a valid labeling, we

may recover a labeling for G as follows. Let $C_j \in \mathrm{Comp}(G, \overline{s})$, with the labeling function $\ell_j : V(C_j) \to \{1, \ldots, |V(C_j)|\}$ returned by applying our canonization procedure recursively to $G[C_j]$. Let $h_j := |C_1| + \cdots + |C_{j-1}|$. We will recover a canonical labeling $\ell : V(G) \to [n]$ by, for each such j and $v \in C_j$, setting $\ell(v) := \ell_j(v) + h_j$. As each vertex of G appears in exactly one C_j, ℓ is well-defined.

We stress here again that the recursive calls to the canonization procedure track the depth to ensure that we do not make $\geq 3 \cdot (\log(n) + 1)$ recursive calls. If the depth parameter is ever larger than $3 \cdot (\log(n) + 1)$, then the algorithm returns $\perp$ to indicate an error. In the recombine stage of our divide and conquer procedure, if any of the labelings returned for the components of $\mathrm{Comp}(G, \overline{s})$ are $\perp$, then the algorithm simply returns $\perp$. Thus, a priori, our algorithm may not return a labeling of the vertices. We will prove later (see Lem. 24) that our algorithm actually does return a labeling.

We now give a more precise description of our algorithm and proceed to prove its correctness. We define a canonical labeling $\mathrm{Can}(G)$ of a graph, via a subroutine $\mathrm{Can}(G, d)$. The subroutine $\mathrm{Can}(G, d)$ takes an n-vertex graph G and a depth parameter d, and outputs either a bijection $\lambda : V(G) \to [n]$ or a failure symbol $\perp$. In pseudocode, our canonical labeling subroutine works as follows:

Algorithm 1 $\mathbf{Can}(G, d)$.

Input: A colored graph $G = (V, E, \chi)$ of rank-width $\leq k$, and a parameter d for depth.

1. If $d > 3 \cdot (\log(n) + 1)$, return $\perp$.
2. If $d \leq 3 \cdot (\log(n) + 1)$ and $|V| = 1$, return $\lambda(v) = 1$.
3. Otherwise, if $d \leq 3 \cdot (\log(n) + 1)$ and $|V| > 1$, do the following steps:
4. Run $(10k + 3, O(\log n))$-WL on G.
5. In parallel, enumerate all possible flip extensions $\overline{s} = (\overline{a}, \overline{b}, f)$ in lexicographic order, where the order on $(\overline{a}, \overline{b})$ is considered with respect to the ordering induced by the coloring $\chi_{10k+3,O(\log n)}$ (by [25], the colors are represented by numbers, and so color class order is well-defined).
6. For each flip extension, $\overline{s} = (\overline{a}, \overline{b}, f)$,
   a. Compute the coloring $\chi^{(\overline{a},\overline{b})}_{10k+3,O(\log(n))}$ applied to G.
   b. Construct the flipped graph $G^{\overline{s}}$.
   c. Compute the set of connected components $\mathrm{Comp}(G, \overline{s})$. If $G^{\overline{s}}$ is connected, then return $\perp$. Note that there exists a rank decomposition (T, γ) in which for all $u, v \in V(T)$, $\gamma(u) \neq \gamma(v)$. So there exists a flip extension $\overline{s}$ that splits $G^{\overline{s}}$ into at least two connected components.
   d. Order the components $C \in \mathrm{Comp}(G, f)$ by lexicographic ordering of the multiset of colors $\chi^{\overline{a},\overline{b}}_{10k+3,O(\log(n))}(G[C])$. Let $C_1, \ldots, C_\ell$ be the components in this ordering.
   e. Compute $\mathrm{Can}(d+1, G[C_1]), \ldots, \mathrm{Can}(d+1, G[C_\ell])$ and let $\lambda_{\overline{s},1}, \ldots, \lambda_{\overline{s},\ell}$ be the resulting labelings.
   f. If $\lambda_{\overline{s},i} = \perp$ for any $i \in [\ell]$ set $\lambda_{\overline{s}} = \perp$. Otherwise, if $\lambda_{\overline{s},1}, \ldots, \lambda_{\overline{s},\ell}$ are the (canonical) labelings returned by the recursive calls, set

$$\lambda_{\overline{s}}(v) = \lambda_{\overline{s},i}(v) + \sum_{j=1}^{i-1} |C_j|$$

   where $C_i \ni v$.
7. Return the labeling $\lambda_{\overline{s}}$ corresponding to the first flip extension $\overline{s}$ (relative to the order in which the flip extensions were enumerated) that is not $\perp$.

We then define the canonical labeling by setting $\mathrm{Can}(G) = \mathrm{Can}(G, 0)$ via the subroutine. We now show that our subroutine satisfies the desired properties.

▶ **Lemma 24.** *If G is a graph of rank-width at most k, the above procedure terminates and does not return $\perp$.*

Proof. For termination, we observe that at each step the depth parameter d increases and that if d becomes larger than $3\log(n) + 1$, the procedure returns. Hence, the procedure must terminate.

We will now show by induction that Algorithm 1 returns a labeling instead of $\perp$. Fix (T, γ) to be a rank decomposition of G, of width $\leq 2k$ and height $\leq 3 \cdot (\log(n) + 1)$. Let $t \in V(T)$, and let t_1, t_2 be the children of t in T. We will use Lem. 11, which provides that for each $t \in V(T)$, there exists a flip extension $\overline{s}$ so that for every $C \in \mathrm{Comp}(G[\gamma(t)], \overline{s})$, there exists an $i = 1, 2$ such that $C \subseteq \gamma(t_i)$. We will use this to show that the algorithm constructs a non-empty set of labelings for G. As the algorithm chooses the least such labeling[4], it follows that the algorithm in fact returns a labeling. Note that while the algorithm will not be explicitly constructing (T, γ), the algorithm still descends along (T, γ) in one of its parallel computations.

Consider first the case when $|V(G)| = 1$. Here, the algorithm returns $\lambda(v) = 1$, where $v \in V(G)$. Now fix a node $t \in V(T)$, and let $\gamma(t)$ be the corresponding set of vertices. Suppose that $|\gamma(t)| > 1$. Let t_1, t_2 be the children of t in T. By Lem. 11, there exists a flip extension $\overline{s} = (\overline{a}, \overline{b}, f)$ such that for every component $C \in \mathrm{Comp}(G[\gamma(t)], \overline{s})$, either $C \in \gamma(t_1)$ or $C \in \gamma(t_2)$. As we consider all flip extensions of $\gamma(t)$ in parallel, one of our parallel computations will consider $\overline{s}$. We will analyze this parallel computation.

Prior to recursively invoking the algorithm on each $G[C]$ ($C \in \mathrm{Comp}(G[\gamma(t)], \overline{s})$), the algorithm first sorts said components (For the purposes of showing that the algorithm yields a (not necessarily canonical) labeling, the precise ordering does not matter. We will argue later that the ordering used by the algorithm is canonical– see Lem. 25). For each $C \in \mathrm{Comp}(G[\gamma(t)], \overline{s})$, the algorithm is then applied recursively to $G[C]$.

Now for $i = 1, 2$, let $C_{i,1}, \ldots, C_{i,j_i} \in \mathrm{Comp}(G[\gamma(t)], \overline{s})$ be precisely the components in $\gamma(t_i)$. Observe that a flip extension on $\gamma(t_i)$ restricts to a flip extension on an individual component $C_{i,h}$ ($h \in [j_i]$). Conversely, given flip extensions $\overline{s}_{i,h}$ ($h \in [j_i]$), the union of these flip extensions induce a flip extension $\overline{s}$ on $\gamma(t_i)$.

By Lem. 11, there exists a flip extension $\overline{s}_i$ such that for every component $C' \in \mathrm{Comp}(G[\gamma(t_i)], \overline{s_i})$, $C' \in \gamma(t_{i,1})$ or $C' \in \gamma(t_{i,2})$. Suppose that s_i is the union of the flip extensions $(\overline{s}_{i,h})_{h \in [j_i]}$. As, for each $h \in [j_i]$, the recursive call of the algorithm applied to $C_{i,h}$ will consider all flip extensions of $C_{i,h}$ in parallel. Thus, via the recursive calls to the components $C_{i,h}$ ($h \in [j_i]$), the algorithm will consider all flip extensions of $\gamma(t_i)$, including the flip extension $\overline{s_i}$. Thus, some parallel choice will descend along (T, γ), and so we may assume that the algorithm computes a labeling for each $C \in G([\gamma(t)], \overline{s})$. As these components are disjoint and listed in a fixed order, the algorithm in fact computes a labeling for $\gamma(t)$. The result now follows by induction. ◀

▶ **Lemma 25.** *Let G be a colored graph of rank-width at most k and let H be an arbitrary graph. If $\lambda : V(G) \to [n]$ and $\kappa : V(H) \to [n]$ are the labelings output by Algorithm 1 on G and H respectively, then $G \cong H$ if and only if the map $\kappa^{-1} \circ \lambda$ is an isomorphism.*

[4] with respect to the order in which the flip extensions were enumerated– See Algorithm 1, Line 5

Proof. If $\kappa^{-1} \circ \lambda$ is an isomorphism, then clearly $G \cong H$. We show that if $G \cong H$, then $\kappa^{-1} \circ \lambda$ is an isomorphism. The proof is by induction on the number of vertices in G. Assume that $|V(G)| = 1$ and $G \cong H$. Note that the algorithm returns $\lambda = \kappa = \mathrm{id}$, the identity permutation, on a graph with one vertex. Thus, $\kappa^{-1} \circ \lambda$ is an isomorphism as desired.

Suppose that $|V(G)| > 1$. Let λ be the labeling returned for G and κ the labeling returned for H. Let $(\overline{a}, \overline{b})$ be the split pair the algorithm selects for λ on the initial call (when the algorithm is invoked on G with depth $= 0$). By the algorithm, $\chi_{10k+3,O(\log n)}((\overline{a}, \overline{b}))$ belongs to the minimal color class where a labeling was returned. Let $(\overline{a}', \overline{b}')$ be the corresponding split pair of H selected for κ. Observe[5] that $(10k+3, O(\log(n))$-WL must assign the same color to the tuples $(\overline{a}, \overline{b})$ and $(\overline{a}', \overline{b}')$.

As the algorithm enumerates the flip extensions in lexicographical order, it considers the flip functions in lexicographical order. As the ordering on flip functions does not depend on the choice of split pair, and we have that $G \cong H$, the flip function[6] $f : (2^{\overline{a} \cup \overline{b}})^2 \to [n] \cup \{\bot\}$ selected for G will also be used for H. Write $\overline{s} := (\overline{a}, \overline{b}, f)$ and $\overline{s}' := (\overline{a}', \overline{b}', f)$. The algorithm next computes the flipped graphs $G^{\overline{s}}$ and $H^{\overline{s}'}$. By Lem. 12, we have that: $(G^{\overline{s}}, \chi^{(\overline{a},\overline{b})}_{6k+3,O(\log n)}) \cong (H^{\overline{s}'}, \chi^{(\overline{a}',\overline{b}')}_{6k+3,O(\log n)})$. It follows that $\ell := |\mathrm{Comp}(G, \overline{s})| = |\mathrm{Comp}(H, \overline{s}')|$. Label the components of $\mathrm{Comp}(G, \overline{s})$ as $C_1, \ldots, C_\ell$, and the components of $\mathrm{Comp}(H, \overline{s}')$ as $D_1, \ldots, D_\ell$. Furthermore, by (4), there exists a bijection $\psi : [\ell] \to [\ell]$ such that for all $i \in [\ell]$, $G[C_i] \cong H[D_{\psi(i)}]$. In particular, as we compute $\chi^{(\overline{a},\overline{b})}_{10k+3,O(\log n)}$ at line 6(a), the isomorphism class of $G[C_i] \cong H[D_{\psi(i)}]$ takes into account how $G[C_i]$ connects to the rest of G and how $H[D_{\psi(i)}]$ connects back to the rest of H (see the discussion in the two paragraphs immediately below Lem. 23). As the algorithm sorts the components of $\mathrm{Comp}(G, \overline{s})$ (respectively, $\mathrm{Comp}(H, \overline{s}')$), we may without loss of generality take ψ to be the identity permutation.

By the inductive hypothesis, we may assume that for each $i \in [\ell]$, the algorithm computes a labeling $\ell_i : C_i \to [|C_i|]$, a labeling $\kappa_i : \psi'(C_i)) \to [|C_i|]$, and that $\kappa_i^{-1} \circ \ell_i$ is an isomorphism. Now by construction, if $v \in C_i$, then

$$\lambda(v) = \lambda_i(v) + \sum_{j=1}^{i-1} |C_j|,$$

and κ is defined analogously. As $C_i \cap C_h = \emptyset$ (resp., $D_i \cap D_h = \emptyset$) whenever $i \neq h$, λ and κ are well-defined. Furthermore, as $\kappa_i^{-1} \circ \lambda_i$ is an isomorphism of $G[C_i] \cong H[D_i]$ for each $i \in [\ell]$, $\kappa|^{-1}_{H[D_i]} \circ \lambda|_{G[C_i]}$ is an isomorphism of $G[C_i] \cong H[D_i]$.

Now suppose that v, w belong to different components of $\mathrm{Comp}(G, \overline{s})$. Let $v' := (\kappa^{-1} \circ \lambda)(v)$ and $w' := (\kappa^{-1} \circ \lambda)(w)$. We will show that $vw \in E(G)$ if and only if $v'w' \in E(H)$. By the definition of the flipped graph (see Section 3), we can determine whether $vw \in E(G)$ based on $N(v) \cap (\overline{a} \cup \overline{b})$, $N(w) \cap (\overline{a} \cup \overline{b})$, and $|N(v) \cap [w]_{\equiv_s}|$. All of this information is encoded in $\chi^{(\overline{a},\overline{b})}_{10k+3,O(\log n)}((v, w))$, and $\chi^{(\overline{a}',\overline{b}')}_{10k+3,O(\log n)}((v, w))$. Thus, $vw \in E(G)$ if and only if $v'w' \in E(H)$. It follows that the map $\kappa^{-1} \circ \lambda$ is an isomorphism. The result follows. ◀

Proof of Thm. 20. Let $\lambda : V(G) \to [n]$ be the output of $\mathrm{Can}(G, 3\log(n) + 1)$. Correctness follows from Lem. 24 and Lem. 25. We now establish the complexity. At each recursive call to $\mathrm{Can}(G, d)$, we invoke $(10k+3, O(\log n))$-WL on G, once at line (4), and then in parallel for each flip extension. Our calls to $(10k+3, O(\log n))$-WL are TC^1-computable [25]. We

[5] Full details of this claim will appear in the full version.

[6] We abuse $\overline{a} \cup \overline{b}$ to denote the indices of the vertices as they appear in $(\overline{a}, \overline{b})$.

write down the flipped graph and identify its connected components in L (Lem. 23). So the non-recursive work within a single call to $\mathrm{Can}(G,d)$ is TC^1-computable. $\mathrm{Can}(G,d)$ makes $n^{O(16^k)}$ recursive calls. The height of our recursion tree is $O(\log n)$. The result follows. ◀

5 Logarithmic Weisfeiler–Leman and Treewidth

In the process of our work, we came across a way to modestly improve the descriptive complexity for graphs of bounded treewidth. Our main result in this section is the following.

▶ **Theorem 26.** *The $(3k+6)$-dimensional Weisfeiler–Leman algorithm identifies graphs of treewidth k in $O(\log n)$ rounds.*

In order to prove Thm. 26, we utilize a result of [5] that graphs of treewidth k admit a *binary* tree decomposition of width $\leq 3k+2$ and height $O(\log n)$. With this decomposition in hand, we leverage a pebbling strategy that is considerably simpler than that of Grohe & Verbitsky [25] and improves the descriptive complexity (Cor. 6).

▶ **Lemma 27.** *Let G, H be graphs. Suppose that a separator $S \subseteq V(G)$ has been pebbled. If the corresponding pebbled set $S' \subseteq V(H)$ is not a separator of H, then Spoiler can win with 3 additional pebbles and $O(\log n)$ additional rounds.*

This next lemma states that if we have pebbled the vertices of some node $\beta(t)$ of the tree decomposition (T,β), then Spoiler can force Duplicator to preserve a given subtree T' (setwise) of the tree decomposition by pebbling some vertex $v \in V(G)$ where there exists $u \in V(T')$ such that $v \in \beta(u)$.

▶ **Lemma 28.** *Let G be a connected graph, and let (T,β) be the binary tree decomposition of G afforded by [5]. Let $t \in V(T)$, and suppose that each vertex in $\beta(t)$ has been pebbled. Let C be the connected component of $T - tu$ that contains u, and let $T' := C \cup tu$.*

Let $v, w \in V(G)$ be vertices contained in the subgraph of G induced by T', such that $v, w \notin \beta(t)$. Suppose that $(v,w) \mapsto (v',w')$ are pebbled. Let $f : V(G) \to V(H)$ be Duplicator's bijection. If v', w' belong to different components of $H \setminus f(\beta(t) \setminus \beta(u))$, then Spoiler can win with 1 additional pebble and $O(\log n)$ additional rounds.

Proof Sketch of Thm. 26. Full details will in the full version. If G is not connected, it is easy to see that Spoiler can force Duplicator to play on non-isomorphic components. Thus, without loss of generality, we may assume that G is connected. Let (T,β) be a tree decomposition for G of width $\leq 3k+2$ and height $O(\log n)$, with T a binary tree, as prescribed by [5]. Let s be the root node of T. Spoiler begins by pebbling the vertices of $\beta(s)$, using $\leq 3k+3$ pebbles. Let $f : V(G) \to V(H)$ be Duplicator's bijection. If $G[\beta(s)] \ncong H[f(\beta(s))]$, then Spoiler wins. So suppose that $G[\beta(s)] \cong H[f(\beta(s))]$.

Let ℓ be the left child of s, and r be the right child of s in T. At the next two rounds, Spoiler places a pebble on some vertex of $\beta(\ell) \setminus \beta(s)$ and a pebble on some vertex of $\beta(r) \setminus \beta(s)$. By Lem. 28, Duplicator must select bijections preserving the left and right sub-trees. Necessarily, either the left or right sub-tree is mapped to a non-isomorphic component of H. Without loss of generality, suppose the left sub-tree is mapped to a non-isomorphic component of H. In this case, Spoiler removes the pebble in $\beta(r)$ and all but one pebble of $\beta(s) \setminus \beta(\ell)$.

We may thus iterate on the above argument, starting from ℓ as the root node in our subtree in the tree decomposition. As $G \ncong H$, we will eventually reach a stage (such as when all of $\beta(t)$ is pebbled for some leaf node $t \in V(T)$) where the map induced by the pebbled vertices does not extend to an isomorphism. In the full version, we will carefully show that only $3k+6$ pebbles on the board and $O(\log n)$ rounds suffice. ◀

References

1 V. Arvind and Piyush P. Kurur. Graph isomorphism is in SPP. *Information and Computation*, 204(5):835–852, 2006. doi:10.1016/j.ic.2006.02.002.

2 L. Babai, E. Luks, and A. Seress. Permutation groups in NC. In *STOC 1987*, STOC '87, pages 409–420, New York, NY, USA, 1987. Association for Computing Machinery. doi:10.1145/28395.28439.

3 László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *STOC'16—Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, pages 684–697. ACM, New York, 2016. Preprint of full version at arXiv:1512.03547v2 [cs.DS]. doi:10.1145/2897518.2897542.

4 László Babai. Canonical form for graphs in quasipolynomial time: Preliminary report. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 1237–1246, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3313276.3316356.

5 Hans L. Bodlaender. NC-algorithms for graphs with small treewidth. In J. van Leeuwen, editor, *Graph-Theoretic Concepts in Computer Science*, pages 1–10, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg. doi:10.1007/3-540-50728-0_32.

6 Harry Buhrman and Steven Homer. Superpolynomial circuits, almost sparse oracles and the exponential hierarchy. In R. K. Shyamasundar, editor, *Foundations of Software Technology and Theoretical Computer Science, 12th Conference, New Delhi, India, December 18-20, 1992, Proceedings*, volume 652 of *Lecture Notes in Computer Science*, pages 116–127. Springer, 1992. doi:10.1007/3-540-56287-7_99.

7 Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992. Originally appeared in SFCS '89. doi:10.1007/BF01305232.

8 Bruno Courcelle and Mamadou Kanté. Graph operations characterizing rank-width and balanced graph expressions. In *Graph-Theoretic Concepts in Computer Science, 33rd International Workshop, WG 2007, Dornburg, Germany, June 21-23, 2007. Revised Papers*, pages 66–75, June 2007. doi:10.1007/978-3-540-74839-7_7.

9 Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1):77–114, 2000. doi:10.1016/S0166-218X(99)00184-5.

10 Bireswar Das, Anirban Dasgupta, Murali Krishna Enduri, and I. Vinod Reddy. On nc algorithms for problems on bounded rank-width graphs. *Information Processing Letters*, 139:64–67, 2018. doi:10.1016/j.ipl.2018.07.007.

11 Bireswar Das, Jacobo Torán, and Fabian Wagner. Restricted space algorithms for isomorphism on bounded treewidth graphs. *Information and Computation*, 217:71–83, 2012. doi:10.1016/j.ic.2012.05.003.

12 Samir Datta, Nutan Limaye, Prajakta Nimbhorkar, Thomas Thierauf, and Fabian Wagner. Planar graph isomorphism is in log-space. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 203–214, 2009. doi:10.1109/CCC.2009.16.

13 Heinz-Dieter Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer, 2 edition, 1994. doi:10.1007/978-1-4757-2355-7.

14 Michael Elberfeld and Ken-ichi Kawarabayashi. Embedding and canonizing graphs of bounded genus in logspace. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 383–392, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2591796.2591865.

15 Michael Elberfeld and Pascal Schweitzer. Canonizing graphs of bounded tree width in logspace. *ACM Trans. Comput. Theory*, 9(3), October 2017. doi:10.1145/3132720.

16 Vyacheslav Futorny, Joshua A. Grochow, and Vladimir V. Sergeichuk. Wildness for tensors. *Lin. Alg. Appl.*, 566:212–244, 2019. Preprint arXiv:1810.09219 [math.RT]. doi:10.1016/j.laa.2018.12.022.

17 Joshua A. Grochow and Youming Qiao. Isomorphism problems for tensors, groups, and cubic forms: completeness and reductions. arXiv:1907.00309 [cs.CC], 2019. doi:10.48550/arXiv.1907.00309.

18 Martin Grohe. Isomorphism testing for embeddable graphs through definability. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, STOC '00, pages 63–72, New York, NY, USA, 2000. Association for Computing Machinery. doi:10.1145/335305.335313.

19 Martin Grohe. Fixed-point definability and polynomial time on graphs with excluded minors. *J. ACM*, 59(5), November 2012. doi:10.1145/2371656.2371662.

20 Martin Grohe and Sandra Kiefer. A Linear Upper Bound on the Weisfeiler-Leman Dimension of Graphs of Bounded Genus. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 117:1–117:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2019.117.

21 Martin Grohe and Sandra Kiefer. Logarithmic Weisfeiler-Leman Identifies All Planar Graphs. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 134:1–134:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2021.134.

22 Martin Grohe and Daniel Neuen. Canonisation and definability for graphs of bounded rank width. *ACM Trans. Comput. Log.*, 24(1):6:1–6:31, 2023. doi:10.1145/3568025.

23 Martin Grohe, Daniel Neuen, Pascal Schweitzer, and Daniel Wiebking. An improved isomorphism test for bounded-tree-width graphs. *ACM Trans. Algorithms*, 16(3), June 2020. doi:10.1145/3382082.

24 Martin Grohe and Pascal Schweitzer. Isomorphism testing for graphs of bounded rank width. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1010–1029, 2015. doi:10.1109/FOCS.2015.66.

25 Martin Grohe and Oleg Verbitsky. Testing graph isomorphism in parallel by playing a game. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part I*, volume 4051 of *Lecture Notes in Computer Science*, pages 3–14. Springer, 2006. doi:10.1007/11786986_2.

26 Lauri Hella. Definability hierarchies of generalized quantifiers. *Annals of Pure and Applied Logic*, 43(3):235–271, 1989. doi:10.1016/0168-0072(89)90070-5.

27 Lauri Hella. Logical hierarchies in PTIME. *Information and Computation*, 129(1):1–19, 1996. doi:10.1006/inco.1996.0070.

28 Sang il Oum and Paul Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4):514–528, 2006. doi:10.1016/j.jctb.2005.10.006.

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

30 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

31 Sandra Kiefer, Ilia Ponomarenko, and Pascal Schweitzer. The Weisfeiler–Leman dimension of planar graphs is at most 3. *J. ACM*, 66(6), November 2019. doi:10.1145/3333003.

32 Johannes Köbler, Sebastian Kuhnert, Bastian Laubner, and Oleg Verbitsky. Interval graphs: Canonical representations in logspace. *SIAM Journal on Computing*, 40(5):1292–1315, 2011. doi:10.1137/10080395X.

33 Johannes Köbler, Uwe Schöning, and Jacobo Torán. Graph isomorphism is low for PP. *Comput. Complex.*, 2:301–330, 1992. doi:10.1007/BF01200427.

34 Johannes Köbler and Oleg Verbitsky. From invariants to canonization in parallel. In Edward A. Hirsch, Alexander A. Razborov, Alexei Semenov, and Anatol Slissenko, editors, *Computer Science – Theory and Applications*, pages 216–227, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. doi:10.1007/978-3-540-79709-8_23.

35 Johannes Köbler, Sebastian Kuhnert, and Oleg Verbitsky. On the isomorphism problem for helly circular-arc graphs. *Information and Computation*, 247:266–277, 2016. doi:10.1016/j.ic.2016.01.006.

36 Richard E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, January 1975. doi:10.1145/321864.321877.

37 Leonid Libkin. *Elements of Finite Model Theory*. Springer, 2004. doi:10.1007/978-3-662-07003-1_1.

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

39 Rudolf Mathon. A note on the graph isomorphism counting problem. *Information Processing Letters*, 8(3):131–136, 1979. doi:10.1016/0020-0190(79)90004-8.

40 Daniel Neuen and Pascal Schweitzer. An exponential lower bound for individualization-refinement algorithms for graph isomorphism. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 138–150. ACM, 2018. doi:10.1145/3188745.3188900.

41 Sang-il Oum. Rank-width is less than or equal to branch-width. *Journal of Graph Theory*, 57(3):239–244, 2008. doi:10.1002/jgt.20280.

42 Uwe Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37(3):312–323, 1988. doi:10.1016/0022-0000(88)90010-4.

43 Thomas Thierauf and Fabian Wagner. The isomorphism problem for planar 3-connected graphs is in unambiguous logspace. *Theory Comput. Syst.*, 47(3):655–673, 2010. doi:10.1007/S00224-009-9188-4.

44 Jacobo Torán. On the hardness of graph isomorphism. *SIAM J. Comput.*, 33(5):1093–1108, 2004. doi:10.1137/S009753970241096X.

45 Oleg Verbitsky. Planar graphs: Logical complexity and parallel isomorphism tests. In *STACS 2007*, pages 682–693, Berlin, Heidelberg, 2007. Springer-Verlag. doi:10.5555/1763424.1763505.

46 Fabian Wagner. Graphs of bounded treewidth can be canonized in AC^1. In *Proceedings of the 6th International Conference on Computer Science: Theory and Applications*, CSR'11, pages 209–222, Berlin, Heidelberg, 2011. Springer-Verlag. doi:10.1007/978-3-642-20712-9_16.