Canonizing Graphs of Bounded Tree Width in Logspace

Michael Elberfeld¹ and Pascal Schweitzer²

- 1 RWTH Aachen University, Aachen, Germany
- 2 RWTH Aachen University, Aachen, Germany

— Abstract

Graph canonization is the problem of computing a unique representative, a canon, from the isomorphism class of a given graph. This implies that two graphs are isomorphic exactly if their canons are equal. We show that graphs of bounded tree width can be canonized in deterministic logarithmic space (logspace). This implies that the isomorphism problem for graphs of bounded tree width can be decided in logspace. In the light of isomorphism for trees being hard for the complexity class logspace, this makes the ubiquitous classes of graphs of bounded tree width one of the few classes of graphs for which the complexity of the isomorphism problem has been exactly determined.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Graph Theory

Keywords and phrases algorithmic graph theory, computational complexity, graph isomorphism, logspace, tree width

Digital Object Identifier 10.4230/LIPIcs.STACS.2016.32

1 Introduction

The graph isomorphism problem (ISOMORPHISM) – deciding whether two given graphs are the same up to renaming vertices – is one of the few fundamental problems in NP for which we neither know that it is solvable in polynomial-time nor that it is NP-complete. Since NP-hardness would imply a collapse of the polynomial hierarchy to its second level [4, 24], significant effort has been put into better understanding the graph-theoretic requirements on input graphs that make ISOMORPHISM polynomial-time decidable. A classical result of Bodlaender [3] shows that ISOMORPHISM is in P (deterministic polynomial time) for graphs of bounded tree width [3]. It is also in P for other graph classes like planar graphs [15, 25] and more general graphs with a crossing-free embedding into a fixed surface [12, 13, 21]. Since ISOMORPHISM is hard for NL (nondeterministic logarithmic space) [26], a deeper complexitytheoretic insight behind the polynomial-time algorithms for embeddable graphs is given by the fact that ISOMORPHISM for graphs embeddable into the plane [7] or a fixed surface [10] is in L (deterministic logarithmic space, also called *logspace*). So far, it has been an open question whether for graphs of bounded tree width the isomorphism problem can also be solved in logspace.

Guided by the goal to determine the complexity of the isomorphism problem for graphs of bounded tree width, there has been a sequence of partial results. Bodlaender's algorithm [3], which places ISOMORPHISM for graphs of bounded tree width in P, was first refined to an upper bound in terms of logarithmic-depth circuits with threshold gates (that means, circuits defining the complexity class TC^1) [14] and later improved to use semi-unbounded fan-in Boolean gates (that means, circuits defining the complexity class SAC^1) [6]. Since the chain



© Michael Elberfeld and Pascal Schweitzer; licensed under Creative Commons License CC-BY

33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016). Editors: Nicolas Ollinger and Heribert Vollmer; Article No. 32; pp. 32:1–32:14 Leibniz International Proceedings in Informatics



LIPICS Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

32:2 Canonizing Graphs of Bounded Tree Width in Logspace

 $L \subseteq NL \subseteq SAC^1 \subseteq TC^1 \subseteq P$ is all we know about the relations of these classes, these works leave the question for a logspace approach for graphs of bounded tree width open. Logspace approaches are known for small constant bounds on the tree width. Indeed, Lindell's [18] classical approach to testing isomorphism of trees provides us with a logspace algorithm for graphs of tree width at most 1. This was generalized to graphs of tree width at most 2 [2] and results for graphs without K_5 as a minor [8] apply to graphs of tree width at most 3. Moreover, k-trees, the maximal graphs of tree width k, admit logspace isomorphism tests [1] as well as graphs with a bounded tree depth [5]. While providing us with logspace algorithms for ever larger classes of graphs, the general question remained open.

Results. Our first main result answers the above question in its most general way by showing that the isomorphism problem for graphs of bounded tree width can be solved in logspace. Together with a result of Jenner et al. [16], showing that the isomorphism problem for trees is L-hard, this pinpoints the complexity of the isomorphism problem for graphs of bounded tree width to be L-complete.

▶ **Theorem 1.** For every positive $k \in \mathbb{N}$, the language ISOMORPHISM-TW-k, which contains exactly the pairs of isomorphic graphs of tree width at most k, is complete for L under first-order reductions.

For testing whether two graphs are isomorphic, it is in practice often helpful to perform a two-step approach that first computes a canonical representative for each isomorphism class, called the *canon*, and then declares the two graphs to be isomorphic exactly if their canons are equal (rather than isomorphic). To also be able to construct an *isomorphism* between the input graphs (that means, a bijective function between the vertex sets of given graphs that preserves their edge relations), it is helpful to have additionally access to an isomorphism from the input graphs to their canons. Such an isomorphism to the canon is called a *canonical labeling* of a graph. An isomorphism between the input graphs can be constructed by composing canonical labelings.

For most isomorphism algorithms that have been developed so far, it was possible, with varying amounts of extra effort, to turn them into an algorithm that computes canons and canonical labelings. Hence, deciding ISOMORPHISM and computing canons often have the same known complexity. However, the current situation for graphs of bounded tree width is different: While the approach from [6] puts the isomorphism problem for graphs of bounded tree width into SAC¹, this is not done by providing a canonization procedure. In fact, the best known upper bound for canonizing graphs of bounded tree width uses logarithmic-depth circuits with unbounded fan-in Boolean gates (that means, circuits defining the complexity class AC¹) [27]. Between these classes only the relation SAC¹ \subseteq AC¹ is known. Our second main result clarifies this situation by canonizing graphs of bounded tree width in logspace.

▶ **Theorem 2.** For every $k \in \mathbb{N}$, there is a logspace-computable mapping that turns a graph G with tree width at most k into an isomorphism-invariant encoding of G (a canon) and an isomorphism to it (a canonical labeling).

Techniques

The known logspace approaches for canonizing certain classes of bounded tree width graphs are based on first computing an isomorphism-invariant tree decomposition for the given input graph and then adjusting Lindell's tree canonization approach to canonize the graph with respect to the decomposition. For example, for k-trees [1] an isomorphism-invariant tree

decomposition arises by taking a graph's maximal cliques and their size-k intersections as the bags of the decomposition and connecting two bags based on inclusion. The resulting tree decomposition is both isomorphism-invariant, which is required for a canonization procedure to be correct, and has width k, which enables the application of an extension of Lindell's approach by taking (the constant number of) orderings of the vertices of the bags into account.

Technique 1: Isomorphism-invariant tree decomposition into bags without clique separators. In general, for graphs of tree width at most k, there is no isomorphism-invariant tree decomposition of width k. A simple example of graphs demonstrating this are cycles, which have tree width 2, but no isomorphism-invariant tree decomposition of width 2. We could hope to find an isomorphism-invariant tree decomposition by allowing approximate tree decompositions (that means, allowing an increase of the width to some constant k'). Again, cycles show that such tree decomposition, but an isomorphism-invariant and polynomial-size collection of tree decompositions. However, for all $k' \in \mathbb{N}$, there are graphs of tree width at most 3 for which the smallest isomorphism-invariant collection of tree decompositions of width k' has exponential size. Simple graphs demonstrating this fact are given by forming the disjoint union of n cycles of length n, and adding a vertex that is adjacent to every other vertex.

We work around this problem by considering isomorphism-invariant tree decompositions that may have bags of unbounded size, but with bags that are easier from a graph-theoretic and algorithmic perspective than the original graph. An algorithm developed recently [19] (which refined the time complexity for ISOMORPHISM on graphs of tree width k from Bodlaender's $n^{O(k)}$ bound to $g(k) \cdot n^{O(1)}$ for a function g) applies a technique from Leimer [17] that turns the input graph into its isomorphism-invariant collection of maximal induced subgraphs without clique separators called *maximal atoms*. (In the example above, the maximal atoms are exactly the enriched cycles.) Conceptually, the first step of the proofs of our main results is similar, but produces isomorphism-invariant tree decompositions into bags that are maximal atoms instead of just the isomorphism-invariant collection whose arrangement as a tree highly depends on the order in which subgraphs are considered. While it is sufficient to have an isomorphism-invariant set of potential bags capturing a tree decomposition in order to perform polynomial-time isomorphism tests (see [22]), in order to apply or work towards logspace techniques it is necessary to have an isomorphism-invariant tree decomposition. Our first main technical contribution consists of the graph-theoretic concepts and algorithmic ideas that are needed to compute isomorphism-invariant tree decompositions into maximal atoms for graphs of bounded tree width.

Technique 2: Nested tree decomposition and a quasi-complete isomorphism-based ordering. Lindell's approach [18] for canonizing trees is based on using a weak order on the class of all trees whose incomparable elements are exactly the isomorphic ones, and showing that the order can be computed in logspace. Das, Torán, and Wagner [6] extended this to also work for graphs with respect to given tree decompositions of bounded width. This is done by adding the idea that, for bounded width, it is possible in logspace to guess partial isomorphisms between bags and recursively check whether they can be extended to isomorphisms between the whole graphs and the tree decompositions. When working with the tree decompositions into maximal atoms described above, it is not possible to just guess and check partial isomorphisms between bags since they have an unbounded width.

32:4 Canonizing Graphs of Bounded Tree Width in Logspace

In order to handle the width-unbounded bags of the above decomposition, we use the fact that (as shown in [19]), after appropriate preprocessing, the maximal atoms have polynomial-size isomorphism-invariant families of approximate tree decompositions. To compute these families, we combine an approach for constructing separator-based tree decompositions from [9] to work with the isomorphism-invariant separators from [19]. If we choose a bounded width tree decomposition for each atom, and replace each atom by the chosen tree decomposition, we can turn the width-unbounded decomposition into a width-bounded decomposition for the whole graph. However, since each maximal atom may be associated with several decompositions, we need to consider for each atom a family of decompositions. We call the structure that is obtained a *nested tree decomposition*. In order to extend the approach that canonizes with respect to width-bounded decompositions to nested tree decompositions, we incorporate a bag refinement step into the weak ordering. It turns root bags of unbounded width into width-bounded tree decompositions. For each candidate tree decomposition of the root bag this triggers a modification of the original tree decomposition. However, it turns out that determining whether there is an isomorphism between two graphs that respects two given nested tree decompositions is as hard as the general graph isomorphism problem. Having a polynomial-time algorithm for this, let alone a logspace algorithm, would thus put the general graph isomorphism problem into P. Consequently, we do not generalize the idea of using isomorphism-based orderings with respect to decompositions in a direct way to nested tree decompositions. Instead, we define an approximation of the isomorphism-based ordering. This approximation has the property that it is isomorphism-invariant (that means, graphs that are isomorphic with respect to given nested decompositions are incomparable) but is only quasi-complete, by which we mean that graphs that are incomparable must be isomorphic but not necessarily via an isomorphism that respects the nested decompositions. Developing the notion of nested tree decompositions along with just the right notion of a quasi-complete isomorphism-based ordering is our second main technical contribution.

Technique 3: Recursive logspace algorithm implementing the quasi-complete ordering. Trying all choices of a decomposition on all of the atoms yields exponentially many refined decompositions in total. Avoiding this exponential blowup, our third main technical contribution is a dynamic-programming approach along the tree decomposition that shows how to cycle through candidate decompositions of the maximal atoms while, still, canonizing the graph along the coarser tree decomposition in logspace.

Since recursively cycling through tree decompositions of a bag needs space, we cannot just use the polynomial-size family of tree decompositions that we get from applying the results of [9] to those of [19] as described above. In order to implement the recursion in logspace, we compute nested tree decompositions that satisfy a certain additional property, which we call *p*-boundedness. It allows us to maintain a trade-off between the number of candidate tree decompositions chosen for each bag and the size of the subdecomposition sitting below the bag. This makes a recursive algorithm that uses only logarithmic space possible.

Organization. Section 2 provides background on graphs and logspace. The remaining paper is structured along the proofs of the theorems: Section 3 shows how to compute isomorphism-invariant decompositions into clique-separator-free graphs, while Section 4 contains the decomposition approach for graphs without clique separators. Section 5 defines the notion of nested decompositions and a weak ordering defined along them, while Section 6 proves that the ordering is logspace-computable for width-bounded and *p*-bounded decompositions.

Section 7 proves the main theorems and Section 8 concludes the paper. Due to lack of space, proofs are often sketched or omitted; see the paper's preprint for details [11].

2 Background

The present section sketches the paper's background on graphs, the isomorphism problem, and logspace.

We denote the set of natural numbers, which start at 0, by \mathbb{N} , and use shorthands $[n,m] := \{n,\ldots,m\}$ and [m] := [1,m] for every $n \in \mathbb{N}$ and $m \in \mathbb{N} \setminus \{0\}$.

For a graph G = (V, E) with vertices V and edges $E \subseteq V \times V$, we define V(G) := V and E(G) := E. All graphs considered in the present paper are finite, undirected and simple (neither parallel edges nor loops are present). We denote the class of all finite graphs by \mathcal{G} . To simplify later definitions, we define the coloring function $\operatorname{col}_G : V(G) \times V(G) \to \mathbb{Z}$ of a graph G as follows. $\operatorname{col}_G(u, v)$ equals -1 if v = w, 1 if $v \neq w$ and $\{u, v\} \in E(G)$, and 0 if $v \neq w$ and $\{u, v\} \notin E(G)$. If G's vertices or edges are colored, we extend the coloring function to return natural number encodings of colors. We use standard definitions related to connectivity functions and tree decompositions. We write a tree decomposition as a tuple $D = (T, \mathcal{B})$ where T is the tree underlying the decomposition and \mathcal{B} is the family of bags that implicitly defines the decomposition's adhesion sets and torsos.

An isomorphism from a (colored) graph G to a (colored) graph H is a bijective mapping $\varphi: V(G) \to V(H)$, such that $\operatorname{col}_G(u, v) = \operatorname{col}_H(\varphi(u), \varphi(v))$ holds for every $u, v \in V(G)$. Graphs G and H that admit an isomorphism between them are isomorphic. This gives rise to an equivalence relation that partitions \mathcal{G} into isomorphism classes. The graph isomorphism problem is the language ISOMORPHISM := { $(G, H) \in \mathcal{G} \times \mathcal{G} \mid G$ and H are isomorphic}. A mapping inv that associates an object inv(G) with every graph $G \in \mathcal{G}$, for example a tree decomposition or a family of tree decompositions, is isomorphism-invariant if for every isomorphism φ between two graphs the result of applying φ and inv is independent of the order in which they are applied. That means, for every isomorphism φ from a graph G to a graph H, replacing all occurrences of vertices $v \in V(G)$ in $\operatorname{inv}(G)$ by their image $\varphi(v)$ yields $\operatorname{inv}(H)$.

Two graphs G and G' are isomorphic with respect to tree decompositions $D = (T, \mathcal{B})$ and $D' = (T', \mathcal{B}')$, respectively, if there exists an isomorphism φ from G to G' and an isomorphism ψ from T to T' satisfying $B'_{\psi(n)} = \{\varphi(v) \mid v \in B_n\}$ for every node $n \in V(T)$. Under these conditions we say that φ respects D and D'. Based on this definition and the way of how it refines the isomorphism equivalence relation among graphs, we also consider canons of graphs with respect to tree decompositions.

A deterministic Turing machine whose working space is logarithmically bounded by the input length is called a *logspace* DTM. The functions $f: \{0,1\}^* \to \{0,1\}^*$ computed by such machines are *logspace-computable* (or *in logspace*). The complexity class L, called *(deterministic) logspace*, contains all *languages* $P \subseteq \{0,1\}^*$ whose characteristic functions are in logspace.

3 Decomposing Graphs into Parts Without Clique Separators

A *clique* is a graph with an edge between every two vertices, including the empty graph by definition. A separation (A, B) is a *clique separation* with *clique separator* $A \cap B$ in a graph G if it (1) separates two vertices $x, y \in V(G)$, and (2) $G[A \cap B]$ is a clique.

We construct isomorphism-invariant tree decompositions for graphs of bounded tree width whose bags induce subgraphs without clique separators and whose adhesion sets are

32:6 Canonizing Graphs of Bounded Tree Width in Logspace

cliques (that means, the torsos are exactly the subgraphs induced by the bags). These tree decompositions serve as an intermediate decomposition step in the proofs of the main theorems.

▶ Lemma 3. For every $k \in \mathbb{N}$, there is a logspace-computable and isomorphism-invariant mapping that turns a graph G with tree width at most k into a tree decomposition D for G in which (1) subgraphs induced by the bags do not contain clique separators, and (2) adhesion sets are cliques.

The tree decomposition we construct to prove the lemma is a refined version of a decomposition of Leimer [17] of graphs into their collections of maximal induced subgraphs without clique separators. The crucial point is that we need to adjust his method to not only output the collection of maximal induced subgraphs without clique separators, which suffices for its application in [19], but also an isomorphism-invariant tree decomposition that is based on it. In order to do that, we replace the approach of [17], which is based on finding clique-separator-free parts in a single phase via computing elimination orderings, by an approach that consists of k+1 steps, where $k \in \mathbb{N}$ is the (constant) tree width of the input graph: Given a connected graph G, step 1 finds the maximal induced subgraphs of Gthat do not contain clique separators of size at most 1. Then we build a tree decomposition whose bags are the computed subgraphs without clique separators of size 1 and adhesion sets are the computed size-1 clique separators. Each following step $c \in \{2, \ldots, k+1\}$ continues in a similar way: We already know (from the previous step) that the subgraphs induced by the bags do not contain clique separators of size at most c-1. For each bag, we find the clique separators of size at most c in its induced graph, compute a tree decomposition whose bags are the induced subgraphs without size-c clique separators and adhesion sets are size-c clique separators. In order to proceed with a single tree decomposition that satisfies the above mentioned precondition, we merge the tree decompositions for the bags into the already computed decomposition. This results in a tree decomposition whose bags induce graphs without clique separators of size at most c and adhesions are clique separators of size at most c. Since graphs of tree width at most k contain only cliques of size at most k+1, step k+1finishes with a tree decomposition whose bags induce graphs without clique separators and adhesion sets induce cliques. Implementing this approach both in an isomorphism-invariant and logspace-computable way requires to refine the classical connection between clique separators and candidate tree representations of chordal completions of a graph in terms of size-bounded clique separators and uniquely-defined tree representations of graphs that arise by a refined notion of chordal completions.

4 Decomposing Graphs Without Clique Separators

The decomposition procedure from the previous section provides us with a tree decomposition whose bags are clique-separator-free. In the present section, we decompose clique-separatorfree graphs further into isomorphism-invariant tree decompositions of bounded width (formalized by Lemma 4). This needs two additional assumptions that we later meet during the proofs of the main theorems. First, the decomposition is based on two distinguished nonadjacent vertices from the graph. Second, we assume that the given graph is improved as defined next.

Let impr: $\mathcal{G} \to \mathcal{G}$ be the mapping that takes a graph G and adds edges between all vertices $u, v \in V(G)$ with $\kappa(u, v) > \operatorname{tw}(G)$, where $\kappa(u, v)$ is the size of a smallest separator that separates u from v. The improperator *improves* the graph by adding edges to G

based on its tree width. To avoid losing information, we introduce a function $\operatorname{col}_{\operatorname{impr}(G)}$ that colors edges that appear originally in the inputs with a different color than those coming from the improvement. The mapping impr is isomorphism-invariant by definition. Besides this, we use three further properties of the mapping impr. First, the graph we get from applying impr is saturated in the sense that a second application of it does not add new edges. Formally, this means $\operatorname{impr}(G) = \operatorname{impr}(\operatorname{impr}(G))$ for every graph G as proved in [20, Lemma 2.5]. Second, the tree decompositions of a graph G are exactly the tree decompositions of $\operatorname{impr}(G)$. This implies $\operatorname{tw}(G) = \operatorname{tw}(\operatorname{impr}(G))$ and is proved in [20, Lemma 2.6]. Third, the mapping impr is logspace-computable for graphs of bounded tree width. This follows from Reingold's algorithm for UNDIRECTED-REACHABILITY, and the fact that the tree width of a graph bounds the size of the separators we need to consider in order to compute impr.

▶ Lemma 4. For every $k \in \mathbb{N}$, there is a $k' \in \mathbb{N}$ and a logspace-computable and isomorphisminvariant mapping that turns every graph G with a distinguished non-edge $\{u, v\} \notin E(G)$, where G (1) has tree width at most k, (2) does not contain clique separators, and (3) is improved (that means, $G = \operatorname{impr}(G)$), into a width-k' tree decomposition $D = (T, \mathcal{B})$ for G.

The construction of the decomposition is based on recursively splitting the graph into smaller subgraphs using size-bounded and isomorphism-invariant separators. In order to do this, we adapt in a first step the isomorphism-invariant separators from [19] and show their logspace-computability. Then we combine this with a logspace approach for handling the recursion involved in this approach from [9].

5 Isomorphism-Based Ordering of Nested Tree Decompositions

We develop the notion of nested tree decompositions to later combine the decomposition that we get from Lemma 3 with the candidate decompositions we get from Lemma 4. Nested tree decompositions are tree decompositions whose parts are not just bags, but where every bag is associated with a family of tree decompositions for the bag's torso. We use polynomial-size nested tree decompositions to represent exponential-size families of width-bounded tree decompositions that arise by replacing bags with tree decompositions from their families. In order to solve the isomorphism problem with the help of nested tree decompositions, we use a recursively defined weak ordering on pairs of graphs and nested tree decompositions. Incomparable elements in this weak ordering represent isomorphic graphs. In the following we first define nested tree decompositions and, then, the weak ordering for them.

A nested (tree) decomposition $\overline{D} = (T, \mathcal{B}, \mathcal{D})$ for a graph G consists of a tree decomposition (T, \mathcal{B}) for G, and a family $\mathcal{D} = (\mathcal{D}_n)_{n \in V(T)}$ where every \mathcal{D}_n is a family of tree decompositions $D \in \mathcal{D}_n$ for the torso of n. Normal tree decompositions can be viewed as nested decompositions where \mathcal{D}_n is empty for every $n \in V(T)$. We adjust some terminology that usually applies to tree decompositions for the use with nested decompositions. Let $\overline{D} = (T, \mathcal{B}, \mathcal{D})$ be a nested decomposition. The definition of the width of a bag B_n in a nested decomposition depends on whether \mathcal{D}_n is empty or contains a set of tree decompositions. If $|\mathcal{D}_n| = 0$, we set $\operatorname{tw}(B_n) := |B_n| - 1$ and $\operatorname{tw}(B_n) := \max \{\operatorname{tw}(D) \mid D \in \mathcal{D}_n\}$, otherwise. The width of \overline{D} is $\operatorname{tw}(\overline{D}) := \max \{\operatorname{tw}(B_n) \mid n \in V(T)\}$. The size of \overline{D} is $|\overline{D}| := \sum_{n \in V(T)} (1 + \max \{|D| + 1 \mid D \in \mathcal{D}_n\})$, where $|\mathcal{D}_n| = 0$ implies $\max \{|D| + 1 \mid D \in \mathcal{D}_n\} = 0$. An (unordered) root set M of a nested decomposition $\overline{D} = (T, \mathcal{B}, \mathcal{D})$ is a subset $M \subseteq B_r$ of the root bag B_r in case $|\mathcal{D}_r| = 0$, and (2) every $D \in \mathcal{D}_r$ has a bag B with $M \subseteq B$ in case $|\mathcal{D}_r| > 0$. An ordered root set σ is an ordering of an unordered root set. Refining a nested decomposition $\overline{D} = (T, \mathcal{B}, \mathcal{D})$ with respect to a tree decomposition $D \in \mathcal{D}_r$ for the

32:8 Canonizing Graphs of Bounded Tree Width in Logspace

root $r \in V(T)$ and an ordered root set σ is done as follows. First, we decompose $G[B_r]$ using D. Then, for each child bag B_c of B_r in \overline{D} , we find the highest bag in D that contains the adhesion set $B_{\{r,c\}} = B_r \cap B_c$ and make B_c adjacent to it. A bag of this kind exists since, by definition, D is a tree decomposition of the torso of B_r . We add a new bag containing the elements of σ . This bag is the new root of the obtained decomposition and adjacent to the highest bag in D that contains all elements of σ (in particular, this operation may change which bag of D is highest). The newly constructed nested decomposition is said to be obtained by *refining* \overline{D} and denoted by $\overline{D}_{D,\sigma}$. The size of a nested decomposition decreases when it is refined. That means $|\overline{D}_{D,\sigma}| < |\overline{D}|$ holds. We use this property for proofs by induction.

To be able to distinguish original bags and bags from refining decompositions, we could mark the bags of D, which arise from the refinement step. We circumvent the need to mark the bags by assuming that the bags B_n with empty \mathcal{D}_n are exactly the marked ones. In turn, we require from all nested decompositions \overline{D} we consider that the set of bags B_n with empty \mathcal{D}_n form a connected subtree in \overline{D} containing the root.

▶ **Proposition 5.** The mapping that turns a nested decomposition $\overline{D} = (T, \mathcal{B}, \mathcal{D})$ with decomposition $D \in \mathcal{D}_r$ and an ordered root set σ into $\overline{D}_{D,\sigma}$ is logspace-computable and isomorphism-invariant.

In order to define the isomorphism-based ordering for nested decompositions, we start to review notions related to composed orderings and define an ordering of graphs with given vertex sequences.

Let \prec be a weak ordering on a set M, and $a \equiv a'$ denote that two elements $a, a' \in M$ are incomparable with respect to \prec . That means, neither $a \prec a'$ nor $a' \prec a$ holds. We define the weak ordering on sequences from $M^* := \bigcup_{n \in \mathbb{N}} M^n$ with respect to \prec as follows. We set $a = a_1 \dots a_s \prec a'_1 \dots a'_t = a'$ for $a, a' \in M^*$ if s < t, or s = t and there is an $i \in [s]$ with $a_i \prec a'_i$ while $a_j \equiv a_j$ holds for every $j \in [i-1]$. The weak ordering on tuples from $M_1 \times \dots \times M_k$ with respect to weak orderings \prec_i for sets M_i , respectively, is defined in the same way except that tuples always have the same length. We denote it by $\prec_{(1,\dots,k)}$. We define a weak ordering on finite subsets of M by setting $M_1 \prec M_2$ for two finite $M_1, M_2 \subseteq M$ based on comparing the sequences we get by sorting their elements to be monotonically increasing with respect to \prec .

We write the *concatenation* of sequences σ and τ as $\sigma\tau$. Suppose that (G, σ) and (G', σ') are pairs consisting of graphs G and G' with sequences of vertices $\sigma = v_1 \dots v_s$ and $\sigma' = v'_1 \dots v'_t$ from the respective graphs. We set $(G, \sigma) \prec_{\text{seq}} (G', \sigma')$ if sequence $\operatorname{col}_G(v_1, v_1) \dots$ $\operatorname{col}_G(v_1, v_s) \operatorname{col}_G(v_2, v_1) \dots \operatorname{col}_G(v_s, v_s)$ is smaller than sequence $\operatorname{col}_{G'}(v'_1, v'_1) \dots$ $\operatorname{col}_{G'}(v'_1, v'_t) \operatorname{col}_{G'}(v'_2, v'_1) \dots \operatorname{col}_{G'}(v'_t, v'_1) \dots \operatorname{col}_{G'}(v'_t, v'_t)$ with respect to the (standard) ordering < of \mathbb{N} . We write $(G, \sigma) \equiv_{\text{seq}} (G', \sigma')$ if (G, σ) and (G', σ') are *incomparable* with respect to \prec_{seq} . The ordering \prec_{seq} is logspace-computable by enumerating all pairs of vertices in lexicographic order of the indices.

Graphs G and G' are isomorphic with respect to sequences of vertices $\sigma = v_1 \dots v_s$ and $\sigma' = v'_1 \dots v'_t$ from the respective graphs if s = t and there is an isomorphism φ from G to G' with $\varphi(v_i) = v'_i$ for every $i \in [s]$. We say that φ respects σ and σ' in this case. Based on this definition, we also consider canons of graphs with respect to vertex sequences. Due to the following statement, which we immediately get from the definition, we call \prec_{seq} an isomorphism-based ordering of graphs with vertex sequences.

{

▶ **Proposition 6.** Let G and G' be graphs with sequences of vertices $\sigma = v_1 \dots v_s$ and $\sigma' = v'_1 \dots v'_t$ from the respective graphs.

- ("invariance"-property) If G and G' are isomorphic with respect to σ and σ' , then $(G, \sigma) \equiv_{seq} (G', \sigma').$
- ("quasi-completeness"-property) If $(G, \sigma) \equiv_{seq} (G', \sigma')$, then the (induced subgraphs) $G[\{v_1, \ldots, v_s\}]$ and $G'[\{v'_1, \ldots, v'_t\}]$ are isomorphic with respect to σ and σ' .

We define an ordering of graphs with nested decompositions by recursively ordering the child decompositions and combining this with the root bags. If a root bag has no refining tree decompositions, this is done by trying all possible orderings of the vertices of the bag. If the root bag has refining tree decompositions, this is done by first refining it before going into recursion.

For each child c of the root node r of a nested decomposition $D = (T, \mathcal{B}, \mathcal{D})$, we define a set $\Pi(c)$ of orderings of a vertex set as follows. If $|\mathcal{D}_c| = 0$, then $\Pi(c)$ contains all orderings of the vertices of B_c . If $|\mathcal{D}_c| > 0$, then $\Pi(c)$ is the set of orderings of the adhesion set $B_{\{r,c\}} = B_r \cap B_c$. We use the sequences from $\Pi(c)$ as ordered root sets for the child decomposition of \overline{D} rooted at c.

For all tuples (G, \bar{D}, σ) and (G', \bar{D}', σ') of graphs with nested decompositions and ordered root sets, we define whether $(G, \bar{D}, \sigma) \prec_{dec} (G', \bar{D}', \sigma')$ holds based on a case distinction: "size"-comparison. If $|\bar{D}| < |\bar{D}'|$, or $|\bar{D}| = |\bar{D}'|$ and $|\mathcal{D}_r| < |\mathcal{D}'_{r'}|$, then set $(G, \bar{D}, \sigma) \prec_{dec} (G', \bar{D}', \sigma')$.

- "bag"-comparison. If $|\bar{D}| = |\bar{D}'| = 1$ (which implies $|\mathcal{D}_r| = |\mathcal{D}'_{r'}| = 0$), then set $(G, \bar{D}, \sigma) \prec_{\text{dec}} (G', \bar{D}', \sigma')$ if $(G, \sigma) \prec_{\text{seg}} (G', \sigma')$.
- "recursive"-comparison. If $|\bar{D}| = |\bar{D}'| > 1$, and $|\mathcal{D}_r| = |\mathcal{D}'_{r'}| = 0$, we compare the decompositions recursively. Let c_1, \ldots, c_s be the children of r in \bar{D} with respective child decompositions $\bar{D}_1, \ldots, \bar{D}_s$ and subgraphs G_1, \ldots, G_s . Let c'_1, \ldots, c'_t be the children of r' in \bar{D}' with respective child decompositions $\bar{D}'_1, \ldots, \bar{D}'_t$ and subgraphs G'_1, \ldots, G'_t . Set $(G, \bar{D}, \sigma) \prec_{\text{dec}} (G', \bar{D}', \sigma')$ if the following relation holds, which compares sets of sets that contain tuples to which $\prec_{(\text{dec,seq})}$ applies directly:

$$\{ ((G_i, \bar{D}_i, \tau), (G, \sigma\tau)) \mid \tau \in \Pi(c_i) \} \mid i \in [s] \}$$

$$\prec_{(\text{dec,seq})} \{ \{ ((G'_i, \bar{D}'_i, \tau'), (G', \sigma'\tau')) \mid \tau' \in \Pi(c'_i) \} \mid i \in [t] \}$$

"refinement"-comparison. If $|\bar{D}| = |\bar{D}'| > 1$, and $|\mathcal{D}_r| = |\mathcal{D}'_{r'}| > 0$, then set $(G, \bar{D}, \sigma) \prec_{\text{dec}} (G', \bar{D}', \sigma')$ if $\{(G, \bar{D}_{D,\sigma}, \sigma) \mid D \in \mathcal{D}_r)\} \prec_{\text{dec}} \{(G', \bar{D}'_{D',\sigma'}, \sigma') \mid D' \in \mathcal{D}'_{r'})\}$ holds.

Graphs G and G' are isomorphic with respect to nested decompositions $\overline{D} = (T, \mathcal{B}, \mathcal{D})$ and $\overline{D'} = (T', \mathcal{B'}, \mathcal{D'})$ as well as ordered root sets σ and σ' , respectively, if there exists an isomorphism φ from G to G' that (1) respects the (normal) tree decompositions (T, \mathcal{B}) and $(T', \mathcal{B'})$, (2) respects the sequences σ and σ' , and (3) for every $n \in V(T)$ there is a bijection π_n from \mathcal{D}_n to $\mathcal{D}_{n'}$, such that φ restricted to B_n respects D and $\pi(D)$ for all $D \in \mathcal{D}_n$. Based on how this definition refines the isomorphism equivalence relation among graphs, we consider canons of graphs with respect to nested decompositions. We call \prec_{dec} an isomorphism-based ordering of graphs with nested decompositions, which is justified by the following lemma.

▶ Lemma 7. Let $(G, \overline{D}, \sigma)$ and $(G', \overline{D}', \sigma')$ be tuples consisting of graphs with respective nested decompositions and ordered root sets.

- ("invariance"-property) If G and G' are isomorphic with respect to \overline{D} and \overline{D}' as well as σ and σ' , then $(G, \overline{D}, \sigma) \equiv_{\text{dec}} (G', \overline{D}', \sigma')$.
- ("quasi-completeness"-property) If $(G, \overline{D}, \sigma) \equiv_{\text{dec}} (G', \overline{D}', \sigma')$, then G and G' are isomorphic with respect to σ and σ' .

32:9

32:10 Canonizing Graphs of Bounded Tree Width in Logspace

The ordering \prec_{dec} is defined in order to satisfy the "quasi-completeness"-property, but not a "completeness"-property saying that $(G, \overline{D}, \sigma) \equiv_{\text{dec}} (G', \overline{D}', \sigma')$ implies that G and G'are isomorphic with respect to σ and σ' as well as \overline{D} and \overline{D}' , too. The reason behind this lies in the fact that deciding an ordering of this kind for nested decompositions of a bounded width is as hard as (general) ISOMORPHISM. (This can be proved by a reduction that turns graphs into pairs of independent sets and nested decompositions, which encode the edges).

6 Computing the Ordering for Nested Tree Decompositions in Logspace

We now investigate methods to space-efficiently evaluate the isomorphism-based ordering described in the previous section. The nested decompositions we are working with always have a bounded width. This makes it possible to implement the "recursive"-comparison of the isomorphism-based ordering space-efficiently. If the child decompositions are small enough (more precisely, they are smaller by a constant fraction in comparison to their parent), then it is possible to store a constant amount of information, and in particular to store orderings of the size-bounded root bag, before descending into recursion, without exceeding a desired logarithmic space bound. If there is a large child decomposition, of which there can be only one, then we can use Lindell's classic technique of precomputing the recursive information before storing anything at all. However, for the "refinement"-comparison, a space-efficient approach turns out to be more challenging. In this case, the ordering asks us to compare various refinements of the root bag. Cycling through these refinements as part of a recursive approach requires too much space, even if the number of decompositions is bounded by a polynomial in the size of the root bag. While it is not clear how to remedy this difficulty in general, the nested decompositions we construct in the proofs of our main theorems satisfy an additional technical condition, called p-boundedness below. This makes it possible to find a trade-off between the recursive space requirement and the space required for cycling through the refinements.

Let \bar{D} be a nested decomposition. Consider a bag n with $|\mathcal{D}_n| > 1$. Let c_1, \ldots, c_t be the children of n sorted by monotonically decreasing size of the respecting subdecompositions D_1, \ldots, D_t . If it exists, let $j \in [t]$ be maximal such that $G[A_n]$ with $A_n := (B_n \cap B_{c_1}) \cup \cdots \cup (B_n \cap B_{c_j})$ is a clique, and $|D_j| > |D_{j+1}|$ holds or j = t holds. Otherwise, set j := 0 and $A_n := \emptyset$. We call the children c_1, \ldots, c_j of n the special children and A_n is the attachment clique of the special children. A nested decomposition \bar{D} is p-bounded for a polynomial $p : \mathbb{N} \to \mathbb{N}$ if for every $n \in V(T)$ and non-special child c of n we have $|\mathcal{D}_n| \leq p(|\bar{D}|/|\bar{D}_c|)$. For non-special nodes we use the p-boundedness condition to trade the number of candidate refining decompositions with the size of subdecompositions. This enables an overall space-efficient recursion leading to a proof of the following lemma.

▶ Lemma 8. For every $k \in \mathbb{N}$ and polynomial $p: \mathbb{N} \to \mathbb{N}$, there is a logspace DTM that, on input of graphs G and G' along with respective nested decompositions \overline{D} and \overline{D}' and ordered root sets σ and σ' where \overline{D} and \overline{D}' (1) have width at most k, and (2) are p-bounded, decides $(G, \overline{D}, \sigma) \prec_{dec} (G', \overline{D}', \sigma').$

7 Testing Isomorphism for and Canonizing Bounded Tree Width Graphs

We first show how to compute isomorphism-invariant width-bounded and p-bounded nested decompositions and, then, apply this to prove Theorems 1 and 2.

▶ Lemma 9. For every $k \in \mathbb{N}$, there is a $k' \in \mathbb{N}$, a polynomial $p: \mathbb{N} \to \mathbb{N}$, and a logspacecomputable and isomorphism-invariant mapping that turns every graph G of tree width at most k into a nested decomposition \overline{D} for G that (1) has width at most k', and (2) is p-bounded.

Proof. Instead of the original input graph G, we work with its improved version, which we can compute in logspace since the tree width of G is bounded. Mapping the input graph to its improved version is isomorphism-invariant and the improved version has the same tree decompositions. In the following, we denote the improved version of the input graph by G.

Let $D = (T, \mathcal{B})$ be the isomorphism-invariant tree decomposition we get from G by applying Lemma 3. Since the lemma guarantees that in D the adhesion sets are cliques, the torso of each bag is equal to the bag itself. To turn D into a nested decomposition it thus suffices to find a family of tree decompositions of width at most some constant $k' \in \mathbb{N}$ for each bag. We will apply Lemma 4 to find such a family. Since D decomposes an improved graph and the adhesion sets are cliques, every $G[B_n]$ for $n \in V(T)$ is also improved.

Thus, based on D, we construct a nested decomposition \overline{D} by considering every node n of D and defining an isomorphism-invariant family \mathcal{D}_n of tree decompositions of the bag B_n . If B_n has size at most k+1, we let the family \mathcal{D}_n consist of a single tree decomposition that is just B_n . Note that by this choice, the bag B_n satisfies both the width bounded and the p-boundedness restriction (for every polynomial p with $p(i) \ge 1$ for all $i \in \mathbb{N}$). If the size of B_n exceeds k+1, we would like to apply Lemma 4 to further decompose B_n . However, for the lemma, we need a pair $\{u, v\} \notin E(G)$ in B_n to serve as the root of the decomposition. We cannot simply iterate over all $\{u, v\} \notin E(G)$ in B_n since the result may violate the p-boundedness condition. We proceed as follows: Let c_1, \ldots, c_t be the children of n sorted by decreasing size of the respecting child decompositions D_{c_1}, \ldots, D_{c_t} . If it exists, let $j \in [t]$ be the maximum, such that $G[A_n]$ with $A_n := (B_n \cap B_{c_1}) \cup \cdots \cup (B_n \cap B_{c_i})$ is a clique, and $|D_{c_j}| > |D_{c_{j+1}}|$ holds or j = t holds. Otherwise, set j := 0 and $A_n := \emptyset$. Thus, A_n is the attachment clique of the special children as defined above. We construct a collection of tree decompositions \mathcal{D}_n for B_n based on whether we have j < t or j = t. If j < t, let $m \ge 1$ be the largest integer with $|D_{c_{j+1}}| = |D_{c_{j+m}}|$. By construction, we can find at least one and at most $((k+1)(m+1))^2$ pairs of nonadjacent vertices $\{u, v\}$ in $G[A'_n]$ for $A'_n := A_n \cup (B_n \cap B_{c_{j+1}}) \cup \dots \cup (B_n \cap B_{c_{j+m}}).$

We define \mathcal{D}_n to be the collection of tree decompositions we obtained by applying Lemma 4 to $G[B_n]$ with pairs $\{u, v\}$ of nonadjacent vertices in $G[A'_n]$. We have $|\mathcal{D}_n| \leq ((k+1)(m+1))^2$. This set of decompositions satisfies the *p*-boundedness restriction with the polynomial $p(m) = ((k+1)(m+1))^2$. If j = t, we consider every pair of nonadjacent vertices $\{u, v\}$ in B_n . Again, for every such $\{u, v\}$, we construct a decomposition for G[B] using Lemma 4. We have $1 \leq |\mathcal{D}_n| \leq |B_n|^2$ in this case, satisfying the *p*-boundedness condition, since B_n only has special children. Since the construction of the collections \mathcal{D}_n is isomorphism-invariant, the entire construction is isomorphism-invariant.

Proof of Theorem 1. Given two graphs G and G', by Lemma 9 we can compute in logarithmic space isomorphism-invariant p-bounded nested decompositions \overline{D} and $\overline{D'}$. By Lemma 7, the graphs are isomorphic if and only if there exist ordered root sets σ and σ' with $(G, \overline{D}, \sigma) \equiv_{dec} (G', \overline{D'}, \sigma')$. By Lemma 8, this can be checked in logarithmic space by iterating over all suitable choices of σ and σ' . The L-hardness for every positive $k \in \mathbb{N}$ follows from the L-hardness of the isomorphism problem for trees (connected graphs of tree width at most 1) proved by Jenner et al. [16].

32:12 Canonizing Graphs of Bounded Tree Width in Logspace

Proof of Theorem 2. We use the isomorphism-invariant mapping from Lemma 9 to turn G into a width-bounded and p-bounded nested decomposition $\overline{D} = (T, \mathcal{B}, \mathcal{D})$. The canonical sequence of G's vertices is based on $(G, \overline{D}, \sigma)$ where σ is the empty vertex sequence. In order to compute a canonical sequence with respect to \prec_{dec} in logspace, we repeatedly apply Lemma 8.

If $|\mathcal{D}_r| = 0$, let $\bar{D}_1, \ldots, \bar{D}_s$ be the child decompositions of G containing at least one vertex that is not in σ . We obtain an order on them by defining $\bar{D}_i < \bar{D}_j$ if $\{((G_i, \bar{D}_i, \tau), (G, \sigma\tau)) \mid$ $\tau \in \Pi(c_i) \} \prec_{(\text{dec,seq})} \{ ((G_j, \overline{D}_j, \tau), (G, \sigma \tau)) \mid \tau \in \Pi(c_j) \}.$ Ties are broken arbitrarily, for example by considering the smallest vertex in the child according to the input ordering. For each child \overline{D}_i we compute an ordering $\tau_i \in \Pi(c_i)$ that minimizes $(G_i, \overline{D}_i, \tau_i)$. We recursively create a canonical sequence outputting the canonical sequence of $(G_i, \overline{D}_i, \tau_i)$ for each child in the order of children just defined. If $|\mathcal{D}_r| > 0$, we iterate over all decompositions in \mathcal{D}_r choosing a tuple from $\{(G, \overline{D}_{D,\sigma}, \sigma) \mid D \in \mathcal{D}_B)\}$ that is minimal with respect to \prec_{dec} . Ties are, again, broken based on the input ordering. For computing the canonical sequence we continue recursively on a minimal $(G, D_{D,\sigma}, \sigma)$ only. In order to obtain a canonical sequence, we alter the nested decomposition slightly whenever we go into the recursion using colored edges. More specifically, Lemma 9 constructs D based on two vertices u and v that form a distinguished non-edge. We insert an edge between u and v and color it with a color that does not appear in G (for example, we use -2). In other words, we set $\operatorname{col}_G(u, v) := -2$. This modification is isomorphism-invariant based on the choice of D. The new edge is covered by a bag of D by construction. Inserting the edge only depends on D and, thus, it is stored recursively in an implicit way. The modification has the consequence that distinguished edges are preserved under isomorphism.

To prove that the sequence is canonical, we show that whenever a tie is broken arbitrarily between two options, then the two options are equivalent. There are two situations when a tie can occur: First, assume $\{((G_i, \bar{D}_i, \tau), (G, \sigma \tau)) \mid \tau \in \Pi(c_i)\} \equiv_{(\operatorname{dec}, \operatorname{seq})} \{((G_j, \bar{D}_j, \tau), (G, \sigma \tau)) \mid \tau' \in \Pi(c_j)\}$ for two child decompositions both containing a vertex not in σ . By Lemma 7, there is an isomorphism from the graph induced by the vertices in \bar{D}_i to the graph induced by the vertices in \bar{D}_i fixing σ . This extends to an automorphism of G by fixing all vertices neither in \bar{D}_i nor \bar{D}_j . Since \bar{D} is isomorphism-invariant this automorphism respects \bar{D} therefore mapping \bar{D}_i to \bar{D}_j . Second, assume $(G_i, (\bar{D}_i)_{D,\sigma}, \sigma) \equiv_{(\operatorname{dec},\operatorname{seq})} ((G_j), (\bar{D}_j)_{D',\sigma}, \sigma)$. By Lemma 7, there is an isomorphism from G_i to G_j , which preserves the distinguished edge. It extends to an automorphism of G that fixes all vertices that neither appear in $(\bar{D}_i)_{D,\sigma}$ nor in $(\bar{D}_j)_{D',\sigma}$. Since \bar{D} is isomorphism-invariant, this automorphism of G respects \bar{D} and since the distinguished edge is preserved it maps $(\bar{D}_i)_{D,\sigma}$ to $(\bar{D}_j)_{D',\sigma}$.

8 Conclusion

We showed how to canonize and compute canonical labelings for graphs of bounded tree width in logspace, and this implies that deciding isomorphic graphs and computing isomorphisms can be done in logspace for graphs of bounded tree width. For the proof we first developed a tree decomposition into clique-separator-free subgraphs that is isomorphism-invariant and logspace-computable. Then we showed how to compute, for each bag, an isomorphisminvariant family of width-bounded tree decompositions in logspace. Finally, we combined both decomposition approaches to construct nested tree decompositions and developed a recursive canonization procedure that works on nested tree decompositions.

Deciding ISOMORPHISM for graphs embeddable into the plane [7] or fixed surfaces [10] is in logspace. These graph classes can be described in terms of forbidding fixed minors,

which also holds for classes of graphs with bounded tree width. This opens up the question of whether these logspace results generalize to any class of graphs excluding fixed minors. For these classes polynomial-time ISOMORPHISM procedures are known [23]. Partial results are known for graphs that exclude the minors K_5 or $K_{3,3}$ [8].

— References

- 1 Vikraman Arvind, Bireswar Das, Johannes Köbler, and Sebastian Kuhnert. The isomorphism problem for *k*-trees is complete for logspace. *Information and Computation*, 217:1–11, 2012. doi:10.1016/j.ic.2012.04.002.
- 2 Vikraman Arvind, Bireswar Das, and Johannes Köbler. A logspace algorithm for partial 2-tree canonization. In *Proceedings of the 3rd International Computer Science Symposium* in Russia (CSR 2008), number 5010 in Lecture Notes in Computer Science, pages 40–51. Springer, 2008. doi:10.1007/978-3-540-79709-8_8.
- 3 Hans L Bodlaender. Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. J. Algorithms, 11(4):631–643, 1990. doi:10.1016/0196-6774(90)90013-5.
- 4 R. B. Boppana, J. Hastad, and S. Zachos. Does co-NP have short interactive proofs? Inf. Process. Lett., 25(2):127–132, May 1987. doi:10.1016/0020-0190(87)90232-8.
- Bireswar Das, MuraliKrishna Enduri, and I.Vinod Reddy. Logspace and FPT algorithms for graph isomorphism for subclasses of bounded tree-width graphs. In 9th International Workshop on Algorithms and Computation (WALCOM 2015), volume 8973 of Lecture Notes in Computer Science, pages 329–334. Springer, 2015. doi:10.1007/978-3-319-15612-5_30.
- 6 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.
- 7 Samir Datta, Nutan Limaye, Prajakta Nimbhorkar, Thomas Thierauf, and Fabian Wagner. Planar graph isomorphism is in log-space. In *Proceedings of the 24th Annual IEEE Confer*ence on Computational Complexity (CCC 2009), pages 203–214. IEEE Computer Society, 2009. doi:10.1109/CCC.2009.16.
- 8 Samir Datta, Prajakta Nimbhorka, Thomas Thierauf, and Fabian Wagner. Graph isomorphism for K_{3,3}-free and K₅-free graphs is in log-space. In Proceedings of the 29th Annual IARCS Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2009), volume 4 of LIPIcs, pages 145–156. Schloss Dagstuhl Leibniz-Zentrum für Informatik, 2009. doi:10.4230/LIPIcs.FSTTCS.2009.2314.
- 9 Michael Elberfeld, Andreas Jakoby, and Till Tantau. Logspace versions of the theorems of Bodlaender and Courcelle. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2010)*, pages 143–152. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.21.
- 10 Michael Elberfeld and Ken-ichi Kawarabayashi. Embedding and canonizing graphs of bounded genus in logspace. In Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014), pages 383–392, New York, NY, USA, 2014. ACM. doi:10.1145/2591796.2591865.
- 11 Michael Elberfeld and Pascal Schweitzer. Canonizing graphs of bounded tree width in logspace. CoRR, abs/1506.07810, 2015. URL: http://arxiv.org/abs/1506.07810.
- 12 Ion S. Filotti and Jack N. Mayer. A polynomial-time algorithm for determining the isomorphism of graphs of fixed genus. In *Proceedings of the 12th Annual ACM Symposium* on Theory of Computing (STOC 1980), pages 236–243, New York, NY, USA, 1980. ACM. doi:10.1145/800141.804671.

32:14 Canonizing Graphs of Bounded Tree Width in Logspace

- 13 Martin Grohe. Isomorphism testing for embeddable graphs through definability. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC 2000)*, pages 63–72. ACM, 2000. doi:10.1145/335305.335313.
- 14 Martin Grohe and Oleg Verbitsky. Testing graph isomorphism in parallel by playing a game. In *Proceedings of 33rd International Colloquium on Automata, Languages and Programming* (ICALP 2006), pages 3–14, 2006. doi:10.1007/11786986_2.
- 15 J. E. Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs (preliminary report). In *Proceedings of the 6th Annual ACM Symposium on Theory of Computing (STOC 1974)*, pages 172–184, New York, NY, USA, 1974. ACM. doi:10.1145/800119.803896.
- 16 Birgit Jenner, Johannes Köbler, Pierre McKenzie, and Jacobo Torán. Completeness results for graph isomorphism. J. Comput. Syst. Sci., 66(3):549–566, 2003. doi:10.1016/S0022-0000(03)00042-4.
- 17 Hanns-Georg Leimer. Optimal decomposition by clique separators. *Discrete Mathematics*, 113(1-3):99–123, 1993. doi:10.1016/0012-365X(93)90510-Z.
- 18 Steven Lindell. A logspace algorithm for tree canonization (extended abstract). In Proceedings of the 24th Annual ACM Symposium on Theory of Computing (STOC 1992), pages 400–404, New York, NY, USA, 1992. ACM. doi:10.1145/129712.129750.
- 19 Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. Fixed-parameter tractable canonization and isomorphism test for graphs of bounded treewidth. In Proceedings of the 55th IEEE Symposium on Foundations of Computer Science (FOCS 2014), pages 186–195. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.28.
- 20 Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. Fixedparameter tractable canonization and isomorphism test for graphs of bounded treewidth. *CoRR*, abs/1404.0818, 2014. URL: http://arxiv.org/abs/1404.0818.
- 21 Gary L. Miller. Isomorphism testing for graphs of bounded genus. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC 1980)*, pages 225–235, New York, NY, USA, 1980. ACM. doi:10.1145/800141.804670.
- 22 Yota Otachi and Pascal Schweitzer. Reduction techniques for graph isomorphism in the context of width parameters. In *Proceedings of the 14th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2014)*, pages 368–379, 2014. doi:10.1007/978-3-319-08404-6_32.
- 23 I. N. Ponomarenko. The isomorphism problem for classes of graphs closed under contraction. Journal of Mathematical Sciences, 55(2):1621–1643, 1991. doi:10.1007/BF01098279.
- 24 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.
- 25 Robert Endre Tarjan. A V² algorithm for determining isomorphism of planar graphs. Information Processing Letters, 1(1):32–34, 1971. doi:10.1016/0020-0190(71)90019-6.
- 26 Jacobo Torán. On the hardness of graph isomorphism. *SIAM Journal on Computing*, 33(5):1093–1108, 2004. doi:10.1137/S009753970241096X.
- 27 Fabian Wagner. Graphs of bounded treewidth can be canonized in AC¹. In Proceedings of the 6th International Computer Science Symposium in Russia (CSR 2011), number 6651 in Lecture Notes in Computer Science, pages 209–222. Springer, 2011. doi:10.1007/ 978-3-642-20712-9_16.