Fixed-parameter tractable canonization and isomorphism test for graphs of bounded treewidth[*]

Daniel Lokshtanov [†]   Marcin Pilipczuk [‡]   Michał Pilipczuk [§]   Saket Saurabh[¶]

Abstract

We give a fixed-parameter tractable algorithm that, given a parameter k and two graphs G_1, G_2, either concludes that one of these graphs has treewidth at least k, or determines whether G_1 and G_2 are isomorphic. The running time of the algorithm on an n-vertex graph is $2^{\mathcal{O}(k^5 \log k)} \cdot n^5$, and this is the first fixed-parameter algorithm for GRAPH ISOMORPHISM parameterized by treewidth.

Our algorithm in fact solves the more general *canonization* problem. We namely design a procedure working in $2^{\mathcal{O}(k^5 \log k)} \cdot n^5$ time that, for a given graph G on n vertices, either concludes that the treewidth of G is at least k, or:

- finds in an isomorphic-invariant way a graph $\mathfrak{c}(G)$ that is isomorphic to G;

- finds an isomorphism-invariant *construction term* — an algebraic expression that encodes G together with a tree decomposition of G of width less than k.

Hence, the isomorphism test reduces to verifying whether the computed isomorphic copies or the construction terms for G_1 and G_2 are equal.

1   Introduction

GRAPH ISOMORPHISM is one of the most fundamental graph problems: Given two graphs G_1, G_2, decide whether G_1 and G_2 are *isomorphic*, i.e., there exists a bijection ϕ between $V(G_1)$ and $V(G_2)$ such that $uv \in E(G_1)$ if and only if $\phi(u)\phi(v) \in E(G_2)$. Despite extensive research on the topic, it is still unknown whether the problem can be solved in polynomial time. On the other hand, there are good reasons to believe that GRAPH ISOMORPHISM is not NP-hard either, since NP-hardness of GRAPH ISOMORPHISM would imply a collapse of the polynomial hierarchy [35].

A significant amount of effort has been put into understanding and broadening the spectrum of classes of graphs where polynomial-time isomorphism tests can be designed. Perhaps the most important example is the classic algorithm of Babai and Luks [3, 28], which solves GRAPH ISOMORPHISM on graphs of maximum degree d in time $n^{\mathcal{O}(d)}$. On the other hand, following polynomial-time solvability of GRAPH ISOMORPHISM on planar graphs [20, 21, 22, 38] it has been investigated how more general topological constraints can be exploited to design efficient algorithms for the problem. Isomorphism tests for graphs of genus g working in time $n^{\mathcal{O}(g)}$ were proposed independently by Filotti and Mayer [16] and by Miller [29]. These results were improved by Ponomarenko [32], who gave an $\mathcal{O}(n^{f(|H|)})$ algorithm for graphs excluding a fixed graph H as a minor, for some function f. The result of Ponomarenko implies also that GRAPH ISOMORPHISM can be solved in polynomial time on graphs of constant treewidth. A simple algorithm for graphs of treewidth k running in time $\mathcal{O}(n^{k+4.5})$ was independently given by Bodlaender [5]. Finally, we mention the result of Arnborg and Proskurowski [2], who gave canonical representation of partial 2- and 3-trees.

Recently, Grohe and Marx [18] obtained a structure theorem for graphs excluding a fixed graph H as a topological minor. This theorem roughly states that such graphs can be decomposed along small separators into parts that are either H-minor-free, or of almost bounded degree (in terms of $|H|$). Using previous algorithms for H-minor-free graphs [32] and bounded-degree graphs [3, 28], they managed to show that GRAPH ISOMORPHISM can be solved in $\mathcal{O}(n^{f(|H|)})$ time for H-topological-minor-free graphs, for some function f. Note that this result generalizes both the algorithms for GRAPH ISOMORPHISM on minor free-graphs [32] and on bounded degree graphs [3, 28]. The work of Grohe and Marx constitutes the current frontier of polynomial-time solvability of GRAPH ISOMORPHISM.

Observe that in all the aforementioned results the exponent of the polynomial depends on the considered parameter, be it the maximum degree, genus, treewidth, or the size of the excluded (topological) minor. In the field of parameterized complexity such algorithms are called XP algorithms (for *slice-wise polynomial*), and are often compared to the more efficient FPT algorithms (for *fixed-parameter tractable*), where the running time is required to be of the form $f(k) \cdot n^c$. Here k is the parameter, f is a computable function, and c is a universal constant independent of k. One of the main research directions in parameterized complexity is to consider problems that admit XP algorithms and determine whether they admit an FPT algorithm; we refer to the monographs [15, 17] for more information on parameterized complexity. It is therefore natural to ask which of the the aforementioned results on GRAPH ISOMORPHISM can be improved to fixed-parameter tractable algorithms.

Prior to this work very little was known about such improvements. In particular, the existence of FPT algorithms for GRAPH ISOMORPHISM parameterized by maximum degree, genus or treewidth of the input graph have remained intriguing open problems. A reader familiar with the algorithmic aspects of treewidth might find it surprising that the existence of an FPT algorithm for GRAPH ISOMORPHISM parameterized by treewidth is a difficult problem. The parameter has been studied intensively during the last 25 years, and is quite well-understood. Many problems that are very hard on general graphs become polynomial time, or even linear-time solvable on graphs of constant treewidth. For the vast majority of these problems, designing an FPT algorithm parameterized by treewidth boils down to designing a straightforward dynamic programming algorithm over the decomposition.

For GRAPH ISOMORPHISM this is not the case, even the relatively simple $\mathcal{O}(n^{k+4.5})$ time algorithm of Bodlaender [5] is structurally quite different from most algorithms on bounded treewidth graphs. This might be the reason why GRAPH ISOMORPHISM was one of very few remaining problems of fundamental nature, whose fixed-parameter tractability when parameterized by treewidth was unresolved.

Therefore, fixed-parameter tractability of GRAPH ISOMORPHISM parameterized by treewidth has been considered an important open problem in parameterized complexity for years. This question (and its weaker variants for width measures lower-bounded by treewidth) was asked explicitly in [9, 10, 23, 26, 30, 39], and appears on the open problem list of the recent edition of the monograph of Downey and Fellows [15].

Most of the related work on the parameterized complexity of GRAPH ISOMORPHISM with respect to width measures considers parameters that are always at least as large as treewidth. The hope has been that insights gained from these considerations might eventually lead to settling the main question. In particular, fixed-parameter tractable algorithms for GRAPH ISOMORPHISM has been given for the following parameters: tree-depth [10], feedback vertex set number [26], connected path distance width [30], and rooted tree distance width [39]. Very recent advances by Otachi and Schweitzer [31] give FPT algorithms for parameterizations by root-connected tree distance width and by connected strong treewidth. Even though all these parameters are typically much larger than treewidth, already settling fixed-parameter tractability for them required many new ideas and considerable technical effort. This supports the statement by Kawarabayashi and Mohar [23] that "[...] even for graphs of bounded treewidth, the graph isomorphism problem is not trivial at all".

Our results. In this paper we answer the question of fixed-parameter tractability of GRAPH ISOMORPHISM parameterized by treewidth in the affirmative, by proving the following theorem:

Theorem 1.1. *There exists an algorithm that, given an integer k and two graphs G_1, G_2 on n vertices, in time $2^{\mathcal{O}(k^5 \log k)} \cdot n^5$ either correctly concludes that $\mathbf{tw}(G_1) \geq k$ or $\mathbf{tw}(G_2) \geq k$, or determines whether G_1 and G_2 are isomorphic.*

In fact, we prove a stronger statement, that is, we provide a *canonization* algorithm for graphs of bounded treewidth. This can be defined in two manners. First, following the formalism of Grohe and Marx [18], we show an algorithm that, given G, constructs a canonical graph $\mathfrak{c}(G)$ on the vertex set $\{1, 2, \ldots, |V(G)|\}$ isomorphic to G, such that $\mathfrak{c}(G_1) = \mathfrak{c}(G_2)$ whenever G_1 and G_2 are isomorphic.

Theorem 1.2. *There exists an algorithm that, given an integer k and a graph G on n vertices, in time $2^{\mathcal{O}(k^5 \log k)} \cdot n^5$ either correctly concludes that $\mathbf{tw}(G) \geq k$ or outputs, in an isomorphic-invariant way, a graph $\mathfrak{c}(G)$ that is isomorphic to G, together with a mapping $\phi : V(G) \to V(\mathfrak{c}(G))$ that certifies this isomorphism.*

A second way is through so-called *construction terms*, defined formally in Section 6: They are algebraic expressions encoding construction procedures for graphs of bounded treewidth. Thus, construction terms can be seen as an alternative definition of treewidth via graph grammars (see Lemma 6.2).

Theorem 1.3. *There exists an algorithm that, given a graph G and a positive integer k, in time $2^{\mathcal{O}(k^5 \log k)} \cdot n^5$ either correctly concludes that $\mathbf{tw}(G) \geq k$, or outputs an isomorphism-invariant term $\mathbf{t}$ that constructs G and uses at most k labels. Moreover, this term has bit-length at most $\mathcal{O}(k \log k \cdot n)$.*

The approach to treewidth and tree decompositions via graph grammars and tree automata dates back to the earliest works on this subject, and is the foundation of intensive research on links between treewidth and monadic second-order logic; we refer to a recent monograph of Courcelle and Engelfriet [12] for a more thorough introduction. Unfortunately there is currently no agreed

standard notation for these concepts. In order to ensure clarity of notation, we introduce our own formalism in Section 6.1.

Let us point out that for all the aforementioned classes where GRAPH ISOMORPHISM can be solved in XP time, corresponding XP canonization algorithms were also developed: for bounded-degree graphs by Babai and Luks [3], for H-minor-free graphs by Ponomarenko [32], and for H-topological-minor-free graphs by Grohe and Marx [18].

We also remark that Theorems 1.1 and 1.2 are straightforward corollaries of Theorem 1.3. For Theorem 1.1, we apply the algorithm of Theorem 1.3 to both G_1 and G_2, and verify whether the obtained terms are equal. Similarly, for Theorem 1.2, we may order the vertices of the input graph G according to the order of their appearance in the canonical term. Formal proofs are provided in Section 6.3.

Our techniques. We now sketch the main ideas behind the proof of Theorem 1.3. The starting point is the classic algorithm of Bodlaender [5] that resolves isomorphism of graphs of treewidth k in time $\mathcal{O}(n^{k+4.5})$. Essentially, this algorithm considers all the $(k+1)$-tuples of vertices of each of the graphs as potential bags of a tree decomposition, and tries to assemble both graphs from these building blocks in the same manner using dynamic programming. It turns out that with a slight modification, the algorithm of Bodlaender can be extended to solve the canonization problem as well. We now direct our attention to speeding up the algorithm.

Our idea is the following: If we were able to constrain ourselves to a small enough family of potential bags, then basically the same algorithm restricted to the pruned family of states would work in FPT time. For this to work we need the family to be of size $f(k) \cdot n^c$ for some function f and constant c. Furthermore, we would need an algorithm that given as input the graph G, computes this family in FPT time. Finally, we would need this family of bags to be *isomorphism invariant*. Informally, we want the pruned family of bags to only depend on the (unlabelled) graph G, and not on the labelling of vertices of G given as input. A formal definition of what we mean by isomorphism invariance is given in the preliminaries.

Therefore, the goal is to find a family $\mathcal{B} \subseteq 2^{V(G)}$ of potential bags that is on one hand isomorphism-invariant and reasonably small, and on the other hand it is rich enough to contain all the bags of some tree decomposition of width at most $g(k)$, for some function g. Coping with this task is the main contribution of this paper, and this is achieved in Theorems 3.4, 4.2, and 5.5. The idea that finding an isomorphism-invariant family of potential bags of size $f(k) \cdot n^c$ is sufficient for designing an isomorphism test was first observed by Otachi and Schweitzer [31].[1] However, their approach for proving this initial step is completely different.

The crucial idea of our construction of a small isomorphism-invariant family of bags is to start with the classic 4-approximation algorithm for treewidth given in the Graph Minors series by Robertson and Seymour [33]; we also refer to the textbook of Kleinberg and Tardos [24] for a comprehensive exposition of this algorithm. Since a good understanding of this algorithm is necessary for our considerations, let us recall it briefly.

The algorithm of Robertson and Seymour constructs a tree decomposition of the input graph G in a top-down manner. More precisely, it is a recursive procedure that at each point maintains a separator S of size at most $3k + \mathcal{O}(1)$, which separates the part of G we are currently working with (call it H) from the rest. The output of the procedure is a tree decomposition of H with the set S as the top adhesion. At each recursive call the algorithm proceeds as follows. If S is small, the algorithm adds to S an arbitrarily chosen vertex $u \in V(H)$. If S is large, the algorithm attempts to break S

---

[1]We remark that even though the work of Otachi and Schweitzer was announced almost simultaneously with our work, we believe it should be considered prior to our results. Their results were presented at a Shonan meeting as early as in May 2013 [19].

into two roughly equal-sized pieces using a separator X of size at most $k+1$. The fact that the graph has treewidth at most k ensures that such a separator X will always exist. The new set S', defined as $S \cup \{u\}$ or $S \cup X$, becomes the top-most bag. Below this bag we attach tree decompositions obtained from the recursive calls for instances $(H := G[N[Z]], S := N(Z))$, where Z iterates over the family of vertex sets of the connected components of $H \setminus S'$. Here, $N(Z)$ denotes the set of the neighbors in Z in G, and $N[Z] = Z \cup N(Z)$ is the closed neighborhood of Z. The crucial point is that if S is large (of size roughly $3k$) and X is of size at most $k+1$, then every such component Z will neighbour at most $|S|$ vertices of S', and hence the size of S will not increase over the course of the recursion.

Our high-level plan is to modify this algorithm so that it works in an (almost) isomorphism-invariant manner, and then return all the produced bags as the family $\mathcal{B}$. At a glance, it seems that the algorithm inherently uses two 'very non-canonical' operations: Adding an arbitrarily chosen vertex u and breaking S using an arbitrarily chosen separator X. Especially canonizing the choice of the separator X seems like a hard nut to crack. We circumvent this obstacle in the following manner: We take X to be the union of 'all possible' separators that break S evenly. The problem that now arises is that it is not clear how to bound the number of neighbours in $S' = S \cup X$ that can be seen from a connected component we want to recurse on; recall that we needed to bound this number by $|S|$, in order to avoid an explosion of the bag sizes throughout the course of the algorithm. The crucial technical insight of the paper, proved in Lemma 3.2, is that if one defines 'all possible separators' in a very careful manner, then this bound holds. The proof of Lemma 3.2 relies on a delicate argument that exploits submodularity of vertex separations.

Even if the problem of canonizing the choice of X is solved, we still need to cope with the arbitrary choice of u in case S is small. It turns out that the problem appears essentially only if the set S is very well connected: For every two vertices $x, y \in S$, the vertex flow between x and y is more than k. In other words, the set S constitutes a *clique separator* in the *k-improved graph of G*, i.e., a graph derived from G by making every pair of vertices with vertex flow more than k adjacent. It is known that for the sake of computing tree decompositions of width k one can focus on the k-improved graph rather than the original one (see, e.g., [6] and Lemma 2.7). Therefore, our algorithm will work without any problems provided that the k-improved graph of the input one does not admit any clique separators. The produced tree decomposition has width $2^{\mathcal{O}(k \log k)}$, due to a possibly exponential number of separators breaking S at each step, and is isomorphism-invariant up to the choice of a single vertex from which the whole procedure begins. By running the algorithm from every possible starting point and computing the union of the families of bags of all the obtained decompositions, we obtain an isomorphism-invariant family of potential bags of size $\mathcal{O}(n^2)$. This result is obtained in Theorem 3.4, which summarizes the case when no clique separators are present.

However, the behaviour of clique separators in the graph has been well understood already in the 1980s, starting from the work of Tarjan [36], and studied intensively from a purely graph-theoretical viewpoint. It turns out that all inclusion-wise minimal clique separators of a graph form a tree-like structure, giving rise to so-called *clique minimal separator decomposition*, which decomposes the graph into pieces that do not admit any clique separators. These pieces are often called *atoms*. Most importantly for us, the set of atoms of a graph is isomorphism-invariant, and can be computed in polynomial time [27]. Therefore, in the general case we can compute the clique minimal separator decomposition of the k-improved graph of the input graph, run the algorithm for the case of no clique separators on each atom separately, and finally output the union of all the obtained families. This result is obtained in Theorem 4.2. We refer to the introductory paper of Berry et al. [4] for more information on clique separators.

The family $\mathcal{B}$ given by Theorem 4.2 is essentially already sufficient for running the modified algorithm of Bodlaender [5] on it, and thus resolving fixed-parameter tractability of GRAPH ISOMORPHISM parameterized by treewidth. However, the bags contained in the family $\mathcal{B}$ may be of size as

much as $2^{\mathcal{O}(k \log k)}$. Our canonization algorithm considers every permutation of every candidate bag. Hence, this would result in a double-exponential dependence on k in the running time. In Section 5 we demonstrate how to reduce this dependence to $2^{\mathrm{poly}(k)}$. More precisely, we prove that instead of the original family $\mathcal{B}$, we can consider a modified family $\mathcal{B}'$ constructed as follows: For every $B \in \mathcal{B}$, we replace B with all the subsets of B that have size $\mathcal{O}(k^4)$. Thus, every bag of $\mathcal{B}$ gives rise to $\binom{2^{\mathcal{O}(k \log k)}}{\mathcal{O}(k^4)} = 2^{\mathcal{O}(k^5 \log k)}$ sets in $\mathcal{B}'$, and hence $|\mathcal{B}'| \leq 2^{\mathcal{O}(k^5 \log k)} \cdot |\mathcal{B}|$. However, now every bag of $\mathcal{B}'$ has only $2^{\mathcal{O}(k^4 \log k)}$ possible permutations, instead of a number that is double-exponential in k. In this manner, we can trade a possible explosion of the size of the constructed family for a polynomial upper bound on the cardinality of its members. This trade-off is achieved in Theorem 5.5, and leads to a better time complexity of the canonization algorithm.

Organization of the paper. Section 2 contains preliminaries. Sections 3, 4, 5 contain proofs of Theorems 3.4, 4.2, 5.5, respectively. In Section 6 we utilize Theorem 5.5 to present the canonization algorithm, i.e., to prove Theorems 1.1 and 1.3. In particular, Section 6.1 introduces the formalism of construction terms. In Section 7 we gather concluding remarks. Proofs of lemmas denoted by (♠) are straightforward, and have been moved to the appendix in order not to disturb the flow of the arguments.

2 Preliminaries

In most cases, we use standard graph notation, see e.g. [14].

Separations, separators, and clique separators. We recall here standard definitions and facts about separations and separators in graphs. For two disjoint vertex sets $A, B \subseteq V(G)$, by $E(A, B)$ we denote the set of edges with one endpoint in A and the second endpoint in B.

Definition 2.1 (separation)**.** A pair (A, B) where $A \cup B = V(G)$ is called a *separation* if $E(A \setminus B, B \setminus A) = \emptyset$. The *separator* of (A, B) is $A \cap B$ and the *order* of a separation (A, B) is $|A \cap B|$.

Let $X, Y \subseteq V(G)$ be two not necessarily disjoint subsets of vertices. Then a separation (A, B) is an X-Y *separation* if $X \subseteq A$ and $Y \subseteq B$. The classic Menger's theorem states that for given X, Y, the minimum order of an X-Y separation is equal to the maximum vertex-disjoint flow between X and Y in G. This minimum order is denoted further $\mu(X, Y)$. Moreover, among the X-Y separations of minimum order there exists a unique one with inclusion-wise minimal A, and a unique one with inclusion-wise minimal B. We will call these minimum-order X-Y separations *pushed towards X* and *pushed towards Y*, respectively.

For two vertices $x, y \in V(G)$, by $\mu(x, y)$ we denote the minimum order of a separation (A, B) in G such that $x \in A \setminus B$ and $y \in B \setminus A$. Note that if $xy \in E(G)$ then such a separation does not exist; in such a situation we put $\mu(x, y) = +\infty$. Again, classic Menger's theorem states that for nonadjacent x and y, the value $\mu(x, y)$ is equal to the maximum number of internally vertex-disjoint x-y paths that can be chosen in the graph. The notions of minimum-order separations pushed towards x and y are defined analogously as before. If the graph we are referring to is not clear from the context, we put it in the subscript by the symbol μ.

We emphasize here that, contrary to X-Y separations, in an $x-y$ separation we require $x \in A \setminus B$ and $y \in B \setminus A$, that is, the separator $A \cap B$ cannot contain x nor y. This, in particular, applies to minimum-order $x - y$ separations pushed towards x or y.

It is known that the Ford-Fulkerson algorithm can actually find the aforementioned separations efficiently.

Lemma 2.2. *Given a graph G, two vertex sets $X, Y \subseteq V(G)$, and an integer k, one can in time $\mathcal{O}(k(|V(G)| + |E(G)|))$ find the minimum-order X-Y-separation pushed towards X, or correctly conclude that the minimum order of such separation is greater than k. A similar claim holds for vertices $x, y \in V(G)$ and the minimum-order x-y separations pushed towards x.*

We now move to the notion of a *clique separation*.

Definition 2.3 (**clique separation**). A separation (A, B) is called a *clique separation* if $A \setminus B \neq \emptyset$, $B \setminus A \neq \emptyset$, and $A \cap B$ is a clique in G.

Note that an empty set of vertices is also a clique, hence any separation with an empty separator is in particular a clique separation. We will say that a graph is *clique separator free* if it does not admit any clique separation. Such graphs are often called also *atoms*, see e.g. [27, 4]. From the previous remark it follows that every clique separator free graph is connected.

Tree decompositions. In this paper it is most convenient to view tree decompositions of graphs as rooted. The following notation originates in Marx and Grohe [18], and we use an extended version borrowed from [13].

Let T be a rooted tree and let t be any non-root node of T. The parent of t in T will be denoted by parent(t). A node s is a *descendant* of t, denoted $s \preceq t$, if t lies on the unique path connecting s to the root. We will also say that t is an *ancestor* of s. Note that in this notation every node is its own descendant as well as its own ancestor.

Definition 2.4 (**tree decomposition**). A *tree decomposition* of a graph G is a pair (T, β), where T is a rooted tree and $\beta \colon V(T) \to 2^{V(G)}$ is a mapping such that:

- for each node $v \in V(G)$, the set $\{t \in V(T) \mid v \in \beta(t)\}$ induces a nonempty and connected subtree of T,

- for each edge $e \in E(G)$, there exists $t \in V(T)$ such that $e \subseteq \beta(t)$.

Sets $\beta(t)$ for $t \in V(T)$ are called the *bags* of the decomposition, while sets $\beta(s) \cap \beta(t)$ for $st \in E(T)$ are called the *adhesions*. We sometimes implicitly identify a node of T with the bag associated with it. The *width* of a tree decomposition T is equal to its maximum bag size decremented by one, i.e., $\max_{t \in V(T)} |\beta(t)| - 1$. The *adhesion width* of T is equal to its maximum adhesion size, i.e., $\max_{st \in E(T)} |\beta(s) \cap \beta(t)|$. We define also additional mappings as follows:

$$\gamma(t) = \bigcup_{u \preceq t} \beta(u),$$

$$\sigma(t) = \begin{cases} \emptyset & \text{if t is the root of } T \\ \beta(t) \cap \beta(\text{parent}(t)) & \text{otherwise,} \end{cases}$$

$$\alpha(t) = \gamma(t) \setminus \sigma(t).$$

The *treewidth* of a graph, denoted $\mathbf{tw}(G)$, is equal to the minimum width among all its tree decompositions. Let us remark that in this paper we will be mostly working with graphs of treewidth *less than* k, while most of the literature on the subject considers graphs of treewidth *at most* k. This irrelevant detail will help us avoid clumsy additive constants in many arguments.

If $\mathcal{B} \subseteq 2^{V(G)}$ is a family of subsets of vertices, then we say that $\mathcal{B}$ *captures* a tree decomposition (T, β) if $\beta(t) \in \mathcal{B}$ for each $t \in V(T)$. In this context we often call $\mathcal{B}$ a *family of potential bags*.

6

Graphs of treewidth at most k are known to be k-*degenerate*, that is, every subgraph of a graph of treewidth at most k has a vertex of degree at most k. This in particular implies that an n-vertex graph of treewidth at most k can have at most kn edges.

Let G be a graph and let $S \subseteq V(G)$. We say that a separation (A, B) in G is α-*balanced for S* if $|(A \setminus B) \cap S|, |(B \setminus A) \cap S| \leq \alpha|S|$. The following observation states that graphs of bounded treewidth provide balanced separations of small order.

Observation 2.5 ([7]). *Let G be a graph with $\mathbf{tw}(G) < k$ and let $S \subseteq V(G)$ be a subset of vertices. Then there exists a $\frac{2}{3}$-balanced separation for S of order at most k.*

Improved graph. For a positive integer k, we say that a graph H is k-*complemented* if the implication $(\mu_H(x, y) \geq k) \Rightarrow (xy \in E(H))$ holds for every pair of vertices $x, y \in V(H)$. For every graph G we can construct a k-*improved graph* $G^{\langle k \rangle}$ by having $V(G^{\langle k \rangle}) = V(G)$ and $xy \in E(G^{\langle k \rangle})$ if and only if $\mu_G(x, y) \geq k$. Observe that $G^{\langle k \rangle}$ is a supergraph of G, since $\mu_G(x, y) = +\infty$ for all x, y that are adjacent in G. Moreover, observe that every k-complemented supergraph of G must be also a supergraph of $G^{\langle k \rangle}$. It appears that $G^{\langle k \rangle}$ is k-complemented itself, and hence it is the unique minimal k-complemented supergraph of G.

Lemma 2.6 ([8, 11],♠). *For every graph G and positive integer k, the graph $G^{\langle k \rangle}$ is k-complemented. Consequently, it is the unique minimal k-complemented supergraph of G.*

For completeness we give a proof of Lemma 2.6 in the appendix. The following lemma formally relates tree decompositions of a graph and its improved graph, and states that for the sake of computing a tree decomposition of small width we may focus on the improved graph. The proof (given in the appendix) is basically an application of a simple fact that two vertices x, y with $\mu(x, y) \geq k$ must be simultaneously present in some bag of every tree decomposition of width smaller than k.

Lemma 2.7 (♠). *Let k be a positive integer and let G be a graph. Then every tree decomposition (T, β) of G that has width less than k, is also a tree decomposition of $G^{\langle k \rangle}$. In particular, if $\mathbf{tw}(G) < k$ then $\mathbf{tw}(G) = \mathbf{tw}(G^{\langle k \rangle})$.*

The idea of focusing on the improved graph dates back to the work of Bodlaender on a linear time FPT algorithm for treewidth [6]. Actually, Bodlaender was using a weaker variant of an improved graph, where an edge is added only if the vertices in question share at least k common neighbors. The main point was that this weaker variant can be computed in linear time [6] with respect to the size of the graph. In our work we use the stronger variant, and we can afford spending more time on computing the improved graph.

Lemma 2.8 (♠). *There exists an algorithm that, given a positive integer k and a graph G on n vertices, works in $\mathcal{O}(k^2 n^3)$ time and either correctly concludes that $\mathbf{tw}(G) \geq k$, or computes $G^{\langle k \rangle}$.*

Isomorphisms and isomorphism-invariance. We say that two graphs G_1, G_2 are *isomorphic* if there exists a bijection $\phi \colon V(G_1) \to V(G_2)$, called *isomorphism*, such that $xy \in E(G_1) \Leftrightarrow \phi(x)\phi(y) \in E(G_2)$ for all $x, y \in V(G_1)$. We will often say that some object $\mathcal{D}(G)$ (e.g., a set of vertices, a family of sets of vertices), whose definition depends on the graph, is *isomorphism-invariant* or *canonical*. By this we mean that for any graph G' that is isomorphic to G with isomorphism ϕ, it holds that $\mathcal{D}(G') = \phi(\mathcal{D}(G))$, where $\phi(\mathcal{D}(G))$ denotes the object $\mathcal{D}(G)$ with all the vertices of G replaced by their images under ϕ. The precise meaning of this term will be always clear from the context. Most often, isomorphism-invariance of some definition or of the output of some algorithm

will be obvious, since the description does not depend on the internal representation of the graph, nor uses any tie-breaking rules for choosing arbitrary objects.

We also extend the notion of isomorphism-invariance to *structures*, which are formed by the considered graph G together with some object $\mathcal{E}$ (e.g., a vertex, a set of vertices, a subgraph). Such structures usually represent the graph together with an initial set of choices made by some algorithm, for instance the starting vertex from which the construction of a tree decomposition begins. We say that some object $\mathcal{D}(G, \mathcal{E})$, whose definition is dependant on the structure $(G, \mathcal{E})$, is *invariant under isomorphism of* $(G, \mathcal{E})$, if $\mathcal{D}(G', \mathcal{E}') = \phi(\mathcal{D}(G, \mathcal{E}))$ for any structure $(G', \mathcal{E}')$ such that ϕ is an isomorphism between G and G' that additionally satisfies $\phi(\mathcal{E}) = \mathcal{E}'$. Again, the precise definition of this term will be always clear from the context.

3 The case of no clique separators

Let G be graph and let $S \subseteq V(G)$ be any subset of vertices. The following definition will be a crucial technical ingredient in our reasonings.

Definition 3.1. Suppose that (S_L, S_R) is a partition of S. We say that a separation (A, B) of G is *stable for* (S_L, S_R) if (A, B) is a minimum-order S_L-S_R separation. A separation (A, B) is S-*stable* if it is stable for some partition of S.

Note that $(S_L, V(G))$ and $(V(G), S_R)$ are both S_L-S_R separations, and they have orders $|S_L|$ and $|S_R|$, respectively. Hence, if (A, B) is a separation that is stable for (S_L, S_R), then in particular $|A \cap B| \leq \min(|S_L|, |S_R|)$.

The following lemma will be the main tool for constructing an isomorphism invariant family of candidate bags.

Lemma 3.2. *Let G be a graph, let $S \subseteq V(G)$ be any subset of vertices, and let $\mathcal{F}$ be any finite family of S-stable separations. Define*

$$X := S \cup \bigcup_{(A,B) \in \mathcal{F}} A \cap B.$$

Then, for every connected component of $G \setminus X$, if Z is its vertex set, then $|N(Z)| \leq |S|$.

Proof. We proceed by induction w.r.t. $|\mathcal{F}|$. For $\mathcal{F} = \emptyset$ we have $N(Z) \subseteq X = S$, so the claim is trivial.

Assume then that $\mathcal{F} = \mathcal{F}' \cup \{(A, B)\}$ for some family $\mathcal{F}'$ with $|\mathcal{F}'| < |\mathcal{F}|$, and let X' be defined in the same manner for $\mathcal{F}'$ as X is for $\mathcal{F}$. As (A, B) is S-stable, there is some a partition (S_L, S_R) of S such that (A, B) is a minimum-order S_L-S_R separation. Take arbitrary connected component of $G \setminus X$, let Z be its vertex set, and let $Z' \supseteq Z$ be the vertex set of the connected component of $G \setminus X'$ that contains Z. Since $G[Z]$ is connected and $Z \cap (A \cap B) = \emptyset$, we have that either $Z \subseteq A \setminus B$ or $Z \subseteq B \setminus A$; without loss of generality assume the former.

Let $(C, D) = (V(G) \setminus Z', N[Z'])$. Observe that (C, D) is a separation in G. Moreover, since $Z' \cap S = \emptyset$, then (C, D) is an S-Z' separation, so in particular also a S-Z separation. Furthermore, observe that $C \cap D = N(Z')$, so by the induction hypothesis we obtain $|C \cap D| = |N(Z')| \leq |S|$. We can partition $V(G)$ into 9 parts, where the part a vertex belongs to depends on its membership to $A \setminus B$, $A \cap B$, or $B \setminus A$, and its membership to $C \setminus D$, $C \cap D$, or $D \setminus C$. Let us call these parts $Q_{1,1}, Q_{1,2}, \ldots, Q_{3,3}$, as depicted on Figure 1.

Observe now that $Z \subseteq (A \setminus B) \cap (D \setminus C) = Q_{1,3}$. Since $X \setminus X' \subseteq A \cap B$ and $C \cap D = N(Z') \subseteq X'$, we have that $N(Z) \subseteq (A \cap C \cap D) \cup (Z' \cap A \cap B) = Q_{1,2} \cup Q_{2,2} \cup Q_{2,3}$. On the other hand, by
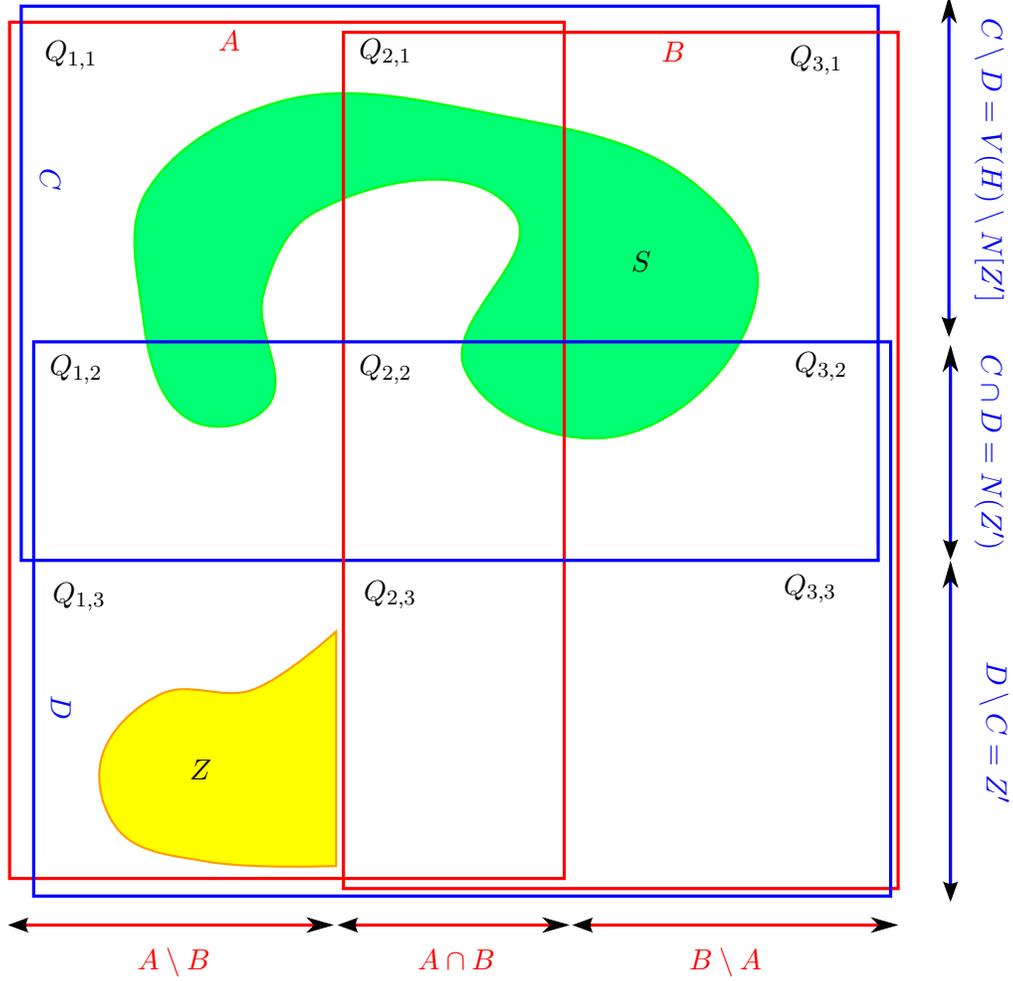
8

Figure 1: Sets in the proof of Lemma 3.2

induction hypothesis we have $|N(Z')| = |Q_{1,2} \cup Q_{2,2} \cup Q_{3,2}| \leq |S|$. Hence, to prove that $|N(Z)| \leq |S|$, it suffices to prove that $|Q_{1,2} \cup Q_{2,2} \cup Q_{2,3}| \leq |Q_{1,2} \cup Q_{2,2} \cup Q_{3,2}|$, or, equivalently, $|Q_{2,3}| \leq |Q_{3,2}|$.

To this end, consider a pair of subsets $(L, R) = (A \cup D, B \cap C) = (V(G) \setminus Q_{3,1}, Q_{2,1} \cup Q_{2,2} \cup Q_{3,1} \cup Q_{3,2})$. We first claim that (L, R) is a separation. Indeed, if $u \in L \setminus R = Q_{1,1} \cup Q_{1,2} \cup Q_{1,3} \cup Q_{2,3} \cup Q_{3,3}$ and $v \in R \setminus L = Q_{3,1}$, then existence of an edge uv would contradict the fact that both (A, B) and (C, D) are separations. Now we claim that (L, R) is an S_L-S_R separation (recall that (A, B) is an S_L-S_R separation by assumption). Indeed, we have $S_L \subseteq A \subseteq L$, and moreover $S_R \subseteq B$ and $S_R \subseteq S \subseteq C$ implies that $S_R \subseteq B \cap C = R$. Since (A, B) is a minimum-order S_L-S_R separation, we have

$$|Q_{2,1} \cup Q_{2,2} \cup Q_{3,2}| = |L \cap R| \geq |A \cap B| = |Q_{2,1} \cup Q_{2,2} \cup Q_{2,3}|.$$

Hence indeed $|Q_{2,3}| \leq |Q_{3,2}|$ and we are done. $\qquad\square$

Lemma 3.2 is used in the following result, which encapsulates one step of the construction of an

9

isomorphism-invariant family of candidate bags. In the sequel, we will use the following parameters:

$$\tau = 6k,$$

$$\rho = \tau + 2(k-1) \cdot \binom{\tau}{2} = \mathcal{O}(k^3),$$

$$\zeta = \rho + 2k \cdot \binom{\rho}{k+1}^2 = 2^{\mathcal{O}(k \log k)}.$$

Lemma 3.3. *Let k be a positive integer and let H be a connected graph that is k-complemented. Let $S \subseteq V(H)$ be a subset of vertices such that (a) $\emptyset \neq S \subsetneq V(H)$, (b) $|S| \leq \rho$, (c) S does* not *induce a clique, (d) $H \setminus S$ is connected, and (e) $S = N_H(V(H) \setminus S)$. There exists an algorithm that either correctly concludes that $\mathbf{tw}(H) \geq k$, or finds a set X with the following properties:*

(i) $X \supsetneq S$, that is, X is a proper superset of S;

(ii) $|X| \leq \zeta$; and

(iii) for every connected component of $H \setminus X$, if Z is its vertex set, then $|N(Z)| \leq \rho$.

The algorithm runs in $2^{\mathcal{O}(k \log k)} \cdot |V(H)|$ time and the constructed set X is invariant with respect to isomorphisms of the structure (H, S).

Proof. We consider two cases. In the first case we assume that $|S| \leq \tau$, and in the second case we assume that $\tau < |S| \leq \rho$.

Case 1: $|S| \leq \tau$. Consider any pair of vertices $x, y \in S$ such that $xy \notin E(H)$. Since H is k-complemented, we have that $\mu_H(x, y) < k$, and hence the minimum-order separations that separate x and y have order less than k. Let $(A_{x,y}^x, B_{x,y}^x)$ and $(A_{x,y}^y, B_{x,y}^y)$ be the minimum-order separations separating x and y that are pushed towards x and y, respectively. Recall that, by the definition of an $x - y$ separation, the separators $A_{x,y}^x \cap B_{x,y}^x$ and $A_{x,y}^y \cap B_{x,y}^y$ do not contain x nor y. We define the set X as follows:

$$X := S \cup \bigcup_{\substack{x,y \in S, \\ xy \notin E(H)}} (A_{x,y}^x \cap B_{x,y}^x) \cup (A_{x,y}^y \cap B_{x,y}^y). \tag{1}$$

In other words, we enhance S by adding, for every pair of nonadjacent vertices from S, both the extreme minimum separators separating them. Observe that the definition of X is invariant with respect to isomorphisms of the structure (H, S). Observe also that $|X| \leq |S| + 2(k-1) \cdot \binom{|S|}{2} \leq \rho$, since $|S| \leq \tau$. This proves properties (ii) and (iii) of X. For property (i), observe that by our assumptions about the set S we have that there exists at least one pair $x, y \in S$ with $xy \notin E(H)$ (by properties (a) and (c)), and that for this pair there exists a path that starts in x, ends in y, and whose all internal vertices are contained in $V(H) \setminus S$ (by properties (a), (d), and (e)). The internal vertices of this path need to include a vertex of $A_{x,y}^x \cap B_{x,y}^x$ and a vertex of $A_{x,y}^y \cap B_{x,y}^y$. Hence, the set X contains at least one vertex outside S, and property (i) holds.

Note that in this case X can be computed in time $k^{\mathcal{O}(1)} \cdot |V(H)|$, by considering every pair of non-adjacent vertices of S, and for each of them running Lemma 2.2 with k as the bound on the order of the separation. (Observe that, unless $\mathbf{tw}(H) \geq k$, we have $|E(H)| = \mathcal{O}(k|V(H)|)$ and, hence, the algorithm of Lemma 2.2 takes $\mathcal{O}(k^2|V(H)|)$ time.)

Case 2: $\tau < |S| \le \rho$. We construct the set X as follows. Consider all the pairs (L, R) of subsets of S such that $L \cap R = \emptyset$ and $|L| = |R| = k+1$. For every such pair, let us verify whether the minimum order of a separation separating L and R is at most k, and in this case let us compute minimum-order L-R separations $(A_{L,R}^L, B_{L,R}^L)$, $(A_{L,R}^R, B_{L,R}^R)$ that are pushed towards L and R, respectively (note that here the vertices of L and R can be included in the separator). We now define X similarly as in the previous case:

$$X := S \cup \bigcup_{\substack{L,R \subseteq S,\ L \cap R = \emptyset, \\ |L|=|R|=k+1,\ \mu(L,R) \le k}} (A_{L,R}^L \cap B_{L,R}^L) \cup (A_{L,R}^R \cap B_{L,R}^R). \tag{2}$$

Note that the definition of X is invariant with respect to isomorphism of the structure (H, S). Property (ii) of X follows directly from the definition and the fact that $|S| \le \rho$. We proceed to checking the other two properties.

For property (iii), take any pair (L, R) considered in the union in (2), and let us look at a separation $(A, B) \in \{(A_{L,R}^L, B_{L,R}^L), (A_{L,R}^R, B_{L,R}^R)\}$. We can easily construct a partition (S_L, S_R) of S such that $L \subseteq S_L \subseteq A$ and $R \subseteq S_R \subseteq B$, by assigning vertices of L to S_L, vertices of R to S_R, and assigning vertices of $S \setminus (L \cup R)$ according to their containment to A or B (thus, we have a unique choice for all the vertices apart from $(S \setminus (L \cup R)) \cap (A \cap B)$). Then (A, B) is an S_L-S_R separation, and since it was a minimum-order L-R separation, it must be also a minimum-order S_L-S_R separation. Hence, (A, B) is an S-stable separation. We infer that all the separations considered in the union in (2) are S-stable. From Lemma 3.2 it follows that $|N(Z)| \le |S| \le \rho$, for every Z being the vertex set of a connected component of $H \setminus X$. This concludes the proof of property (iii).

For property (i), if $\mathbf{tw}(H) < k$ then Observation 2.5 implies an existence of a separation (A, B) of order at most k that is $\frac{2}{3}$-balanced for S. Consequently,

$$|(A \setminus B) \cap S| \ge |S| - |A \cap B| - |(B \setminus A) \cap S| \ge \frac{1}{3}|S| - k > k,$$

where the last inequality follows from the fact that $|S| > \tau = 6k$. Symmetrically, $|(B \setminus A) \cap S| > k$. Let then L and R be any two subsets of $(A \setminus B) \cap S$ and $(B \setminus A) \cap S$, respectively, that have sizes $k+1$. Observe that the existence of separation (A, B) certifies that $\mu(L, R) \le k$, and hence the pair (L, R) is considered in the union in (2). Recall that $(A_{L,R}^L, B_{L,R}^L)$ is then the corresponding L-R separation of minimum order that is pushed towards L. Since $|A_{L,R}^L \cap B_{L,R}^L| \le k$ and $|L|, |R| = k+1$, there exist some vertices x, y such that $x \in L \setminus (A_{L,R}^L \cap B_{L,R}^L)$ and $y \in R \setminus (A_{L,R}^L \cap B_{L,R}^L)$. Similarly as in Case 1, there exists a path that starts in x, ends in y, and whose all internal vertices are contained in $V(H) \setminus S$ (by properties (a), (d), and (e)). The internal vertices of this path need to contain at least one vertex from $A_{L,R}^L \cap B_{L,R}^L$. We infer that if $\mathbf{tw}(H) < k$, then X computed using formula (2) is a proper superset of S. Therefore, we may output the conclusion that $\mathbf{tw}(H) \ge k$ if X computed using formula (2) is equal to S; otherwise X is a proper superset of S, as requested.

Note that in this case X can be computed in time $2^{\mathcal{O}(k \log k)} \cdot |V(H)|$ as follows. There are at most $\binom{\rho}{k+1}^2 = 2^{\mathcal{O}(k \log k)}$ possible choices for L and R, and for each of them we may check whether $\mu(L, R) \le k$ (and compute $(A_{L,R}^L, B_{L,R}^L)$ and $(A_{L,R}^R, B_{L,R}^R)$, if needed) using Lemma 2.2. Similarly as in Case 1, we may assume that the algorithm of Lemma 2.2 takes $\mathcal{O}(k^2 |V(H)|)$ time. $\qquad\square$

Armed with Lemma 3.3, we may proceed to the main result of this section, that is, the enumeration of an isomorphism-invariant family of bags that captures a tree decomposition of reasonably small width.

Theorem 3.4. *Let k be a positive integer, and let G be a graph on n vertices that is clique-separator free (in particular, connected), and k-complemented. There exists an algorithm that computes an isomorphism-invariant family of potential bags $\mathcal{B} \subseteq 2^{V(G)}$ with the following properties:*

(i) $|B| \leq \zeta$ for each $B \in \mathcal{B}$;

(ii) $|\mathcal{B}| = \mathcal{O}(n^2)$;

(iii) assuming that $\mathbf{tw}(G) < k$, the family $\mathcal{B}$ captures some tree decomposition of G that has width at most $\zeta - 1 = 2^{\mathcal{O}(k \log k)}$ and adhesion width at most $\rho = \mathcal{O}(k^3)$.

Moreover, the algorithm runs in $2^{\mathcal{O}(k \log k)} \cdot n^3$ time.

Proof. We first consider the border case when G is a clique. Then we can output $\mathcal{B} = \{V(G)\}$ if $n \leq k$, and $\mathcal{B} = \emptyset$ if $n > k$. In the following we assume that G is not a clique.

Let u be any vertex of G whose degree is less than k. Observe that since graphs of treewidth t are t-degenerate, then there exists at least one such vertex, provided that $\mathbf{tw}(G) < k$ (otherwise we may output $\mathcal{B} = \emptyset$). For each such vertex u we will construct a family $\mathcal{B}_u$ such that (a) $\mathcal{B}_u$ satisfies properties (i) and (iii), (b) $|\mathcal{B}_u| = \mathcal{O}(n)$, (c) the definition of $\mathcal{B}_u$ is invariant with respect to isomorphisms of the structure (G, u), and (d) computing $\mathcal{B}_u$ can be done in time $2^{\mathcal{O}(k \log k)} \cdot n^2$. The final family $\mathcal{B}$ can be then defined simply as:

$$\mathcal{B} := \bigcup_{\substack{u \in V(G) \\ \deg(u) < k}} \mathcal{B}_u.$$

It follows that the definition of $\mathcal{B}$ is isomorphism-invariant, and the running time of the whole algorithm follows from the running time of computing a single $\mathcal{B}_u$.

Let us focus on one vertex u. The algorithm will actually compute some tree decomposition (T_u, β_u) of G that has width $2^{\mathcal{O}(k \log k)}$ and adhesion width $\mathcal{O}(k^3)$, and then it will output the set of all its bags as $\mathcal{B}_u$. Hence it will be clear that (T_u, β_u) is captured by $\mathcal{B}_u$. The definition of (T_u, β_u) will be invariant with respect to isomorphisms of the structure (G, u).

We describe the algorithm as a recursive routine that takes as input an induced subgraph G' of G together with a set X, $\emptyset \neq X \subseteq V(G')$, that is supposed to be the root bag. We ensure that the algorithm does not loop using a potential $\Phi(G', X) = (|V(G')| + 1) \cdot (\zeta + 1) - |X|$. Formally, every call of the algorithm will use only calls for inputs with a strictly smaller potential, and the potential is always nonnegative. For the set X, we require the following invariants:

(a) $|X| \leq \zeta$,

(b) $X \supseteq N_G(V(G') \setminus X)$, that is, X separates $V(G') \setminus X$ from the rest of the graph,

(c) $|N_G(Z)| \leq \rho$ for every connected component of $G' \setminus X$ with vertex set Z, and

(d) $u \notin N_G[V(G') \setminus X]$.

The output of the algorithm will be a tree decomposition of G' that has $\mathcal{O}(|V(G')|)$ bags, width $2^{\mathcal{O}(k \log k)}$ and adhesion width $\mathcal{O}(k^3)$, and whose root bag is equal to X. The top-most call of the routine is for $G' = G$ and $X = N[u]$; that is, we will construct a tree decomposition of G with the top bag being $N[u]$. Note that the invariants (a), (b), (c), and (d) are satisfied in this call.

The algorithm first checks whether $X = V(G')$, in which case it simply returns a decomposition consisting of one root bag being X. Assume then that $X \subsetneq V(G')$. The algorithm considers all the

connected components of $G' \setminus X$. Let Z be the vertex set of one of them; recall that $|N(Z)| \leq \rho$. Let us examine the graph $G''_Z = G'[N[Z]]$, and let $S_Z = N(Z)$. Observe that the pair (G''_Z, S_Z) satisfies the prerequisites of Lemma 3.3:

- prerequisite (a) follows from the fact that Z is nonempty and G is connected;

- prerequisite (b) follows from invariant (c);

- prerequisite (c) follows from the observation that if S_Z induced a clique, then this clique would be a clique separator separating any vertex of Z from u (since invariants (b) and (d) are satisfied);

- prerequisite (d) follows from the fact that Z is connected;

- prerequisite (e) follows from the definition of S_Z.

Hence, let us apply the algorithm of Lemma 3.3 to the pair (G''_Z, S_Z), obtaining a set X_Z. (If the algorithm of Lemma 3.3 returned that $\mathbf{tw}(G''_Z) \geq k$, then also $\mathbf{tw}(G) \geq k$ and we may return $\mathcal{B} = \emptyset$.)

Having computed X_Z, the algorithm calls itself recursively for the graph G''_Z and the top bag X_Z. Note that by the properties guaranteed by Lemma 3.3, we either have that $|V(G''_Z)| < |V(G')|$ or that $G' = G''_Z$ and $|X_Z| > |X|$. Hence we have that $\Phi(G''_Z, X_Z) < \Phi(G', X)$, as was requested. Let us check also that the invariants are satisfied for this call: Invariant (b) follows from the definitions of G''_Z and S_Z, invariants (a) and (c) follow directly from Lemma 3.3, and invariant (d) follows from the definition of (G''_Z, X_Z), and satisfaction of this invariant for the call (G', X).

Let (T_Z, β_Z) be the returned tree decomposition of G''_Z; this decomposition has guaranteed small width and adhesion width, and its top bag is X_Z. We construct the final output decomposition of G' by creating a root bag equal to X, and attaching all the decompositions (T_Z, β_Z) for connected components Z as subtrees below this bag. It is easy to verify that this is indeed a tree decomposition of the graph G'. Moreover, it has required width and adhesion width; note here that adhesions adjacent to the root bag are neighbourhoods of connected components of $G' \setminus X$, which have size at most ρ by invariant (c).

Observe also that since we start with the top-most call $(G, N[u])$, and operations performed in each recursive call (G', X) are invariant with respect to isomorphisms of the structure (G', X), it follows that the computed tree decomposition of G is invariant with respect to isomorphisms of the structure (G', u).

We are left with bounding the number of bags of the returned decomposition and analyzing the running time of the algorithm. Let (T, β) be the returned tree decomposition of G. Recall that for every $v \in V(G)$, the set of nodes whose bags contain v induces a connected subtree of T. Let $t(v)$ be the top-most node of this subtree, i.e., the top-most node where v appears in the bag. We will also say that vertex v *charges* node $t(v)$. We now claim that every node of the decomposition (T, β) is charged at least once. Indeed, the root node with bag $N[u]$ is charged by u, and for every other node, its bag appears as X_Z in some recursive call (G', X). By Lemma 3.3, property (i), we have $X_Z \setminus X \neq \emptyset$, and so the considered node will be charged by any vertex of $X_Z \setminus X$. Consequently, the total number of produced nodes is at most the total number of vertices of the graph, which is n.

To bound the running time of the algorithm, we sum the total work performed in each recursive call of the algorithm. Let us take one call, say (G', X), and observe that the work performed in this call (excluding recursive sub-calls) consists of applications of the algorithm of Lemma 3.3 to instances (G''_Z, S_Z), for all connected components Z of $G' \setminus X$. By Lemma 3.3, each such application (together with auxiliary operations like construction of the subinstance) takes at most $2^{\mathcal{O}(k \log k)} \cdot n$ time, and results in obtaining one subtree of the decomposition that is then attached below the bag X. Let

us assign the time spent on running the algorithm of Lemma 3.3 on instance (G_Z'', S_Z) to the root node of this subtree, i.e., to the node with bag X_Z. Observe that thus every node of the constructed tree decomposition (T, β) of G is being assigned at most once. Since the total number of nodes in T is at most n, we infer that the total time used by the algorithm is $2^{\mathcal{O}(k \log k)} \cdot n^2$, as requested. $\qquad\square$

4   Taming clique separators

The main tool that we will use in this section is a decomposition theorem that breaks the graph using minimal clique separators into pieces that cannot be decomposed further. It appears that such a decomposition can be done, with a set of its bags defined in a unique manner. The idea of decomposing a graph using clique separators dates back to the work of Tarjan [36], and has been studied intensively thereafter. We refer to an introductory article of Berry et al. [4] for more details. The following theorem states all the properties of this decomposition in the language of tree decompositions.

Theorem 4.1 (see e.g. [27, 4])**.** *Let G be a connected graph. There exists a tree decomposition $(T^\star, \beta^\star)$ of G, called* clique minimal separator decomposition, *with the following properties:*

- *for every $t \in V(T^\star)$, $G[\beta^\star(t)]$ is clique-separator free;*

- *each adhesion of $(T^\star, \beta^\star)$ is a clique in G.*

Moreover, $T^\star$ has at most $n-1$ nodes, and the bags of $(T^\star, \beta^\star)$ are exactly all the inclusion-wise maximal induced subgraphs of G that are clique-separator free. Consequently, the family of bags of $(T^\star, \beta^\star)$ is isomorphism-invariant. Finally, the decomposition $(T^\star, \beta^\star)$ can be computed in $\mathcal{O}(nm)$ time.

We remark that the exact shape of the clique minimal separator decomposition is not isomorphism-invariant: The construction procedure depends on the order in which inclusion-wise minimal clique separators of the graph are considered. However, the family of bags of this decomposition is isomorphism-invariant, since these bags may be characterized as all the inclusion-wise maximal induced subgraphs of G that are clique-separator free. In other words, all the possible runs of the decomposition algorithm yield the same family of bags, just arranged in a different manner.

Theorem 4.1 enables us to conveniently extend Theorem 3.4 to graphs that may contain clique separations.

Theorem 4.2. *Let k be a positive integer, and let G be a graph on n vertices that is k-complemented. There exists an algorithm that computes an isomorphism-invariant family of bags $\mathcal{B}$ with the following properties:*

(i) *$|B| \leq \zeta$ for each $B \in \mathcal{B}$;*

(ii) *$|\mathcal{B}| \leq \mathcal{O}(k^2 n^2)$;*

(iii) *assuming that $\mathbf{tw}(G) < k$, the family $\mathcal{B}$ captures some tree decomposition of G that has width at most $\zeta - 1 = 2^{\mathcal{O}(k \log k)}$ and adhesion width at most $\rho = \mathcal{O}(k^3)$.*

Moreover, the algorithm runs in $2^{\mathcal{O}(k \log k)} \cdot n^3$ time.

Proof. In $\mathcal{O}(nm)$ time we compute a clique minimal separator decomposition $(T^\star, \beta^\star)$ of G. We can assume that all the adhesions of $(T^\star, \beta^\star)$ are of size at most k, since otherwise G contains a clique on $k+1$ vertices; then, $\mathbf{tw}(G) \geq k$ and we may output $\mathcal{B} = \emptyset$. In the following, let $r^\star$ be the root of $T^\star$. Observe that:

$$\sum_{t \in V(T^\star)} |\beta^\star(t)| = n + \sum_{t \in V(T^\star) \setminus \{r^\star\}} |\sigma^\star(t)| \leq n + (n-2)k = \mathcal{O}(kn).$$

14

For every $t \in V(T^\star)$, let us examine the graph $G[\beta^\star(t)]$. Since $G[\beta^\star(t)]$ does not admit a clique separation, it is in particular connected. Moreover, since G is k-complemented, then so is $G[\beta^\star(t)]$. Hence, $G[\beta^\star(t)]$ satisfies the prerequisites of Theorem 3.4.

For each $t \in V(T^\star)$, let us apply the algorithm of Theorem 3.4 with parameter k to the graph $G[\beta^\star(t)]$. Let $\mathcal{B}_t$ be the obtained family of bags. We now define the output family to be simply $\mathcal{B} := \bigcup_{t \in V(T^\star)} \mathcal{B}_t$. Observe that since the family of bags of $(T^\star, \beta^\star)$ is isomorphism-invariant, and for each $t \in V(T^\star)$ the constructed family $\mathcal{B}_t$ is invariant with respect to isomorphisms of the graph $G[\beta^\star(t)]$, then it follows that the definition of $\mathcal{B}$ is isomorphism-invariant.

We now verify the requested properties of family $\mathcal{B}$. Property (i) follows directly from the construction and Theorem 3.4. For property (ii), by Theorem 3.4 we have that $|\mathcal{B}_t| \leq \mathcal{O}(|\beta^\star(t)|^2)$ for each $t \in V(T)$. Since $\sum_{t \in V(T^\star)} |\beta^\star(t)| = \mathcal{O}(kn)$, it follows that $|\mathcal{B}| \leq \mathcal{O}(k^2 n^2)$. Note here that a similar argument gives the bound on the running time of the algorithm: Processing graph $G[\beta^\star(t)]$ takes $2^{\mathcal{O}(k \log k)} \cdot |\beta^\star(t)|^3$ time, so the whole algorithm may be implemented in time $\mathcal{O}(nm) + 2^{\mathcal{O}(k \log k)} \cdot k^3 n^3 = 2^{\mathcal{O}(k \log k)} \cdot n^3$.

We are left with verifying property (iii). Assume that $\mathbf{tw}(G) < k$; then also $\mathbf{tw}(G[\beta^\star(t)]) < k$ for each $t \in V(T^\star)$. By Theorem 3.4, for each $t \in V(T^\star)$ there exists some tree decomposition (T_t, β_t) of $G[\beta^\star(t)]$ that is captured by $\mathcal{B}_t$ and satisfies properties (i), (ii), and (iii) of Theorem 3.4. Since $\sigma^\star(t)$ is a clique for each $t \in V(T^\star)$, it follows that some bag of (T_t, β_t) contains the whole $\sigma^\star(t)$. By re-rooting the decomposition (T_t, β_t) if necessary, without loss of generality we may assume that $\beta_t(r_t) \supseteq \sigma^\star(t)$ for each $t \in V(T^\star)$, where r_t is the root of T_t.

Now the goal is to combine all the decompositions (T_t, β_t) into one decomposition (T, β) of G that satisfies the conditions expressed in property (iii). We construct (T, β) by a bottom-up induction on decomposition $(T^\star, \beta^\star)$. For each $t \in V(T^\star)$ we will construct a decomposition (T'_t, β'_t) of $G[\gamma^\star(t)]$ with the property that $\sigma^\star(t)$ will be contained in the root bag of (T'_t, β'_t). Then we will simply take $(T, \beta) := (T'_{r^\star}, \beta'_{r^\star})$.

Assume we are considering a node $t \in V(T^\star)$, and assume that we have constructed decompositions $\{(T'_{t_i}, \beta'_{t_i})\}_{1 \leq i \leq p}$ for the children $t_1, t_2, \ldots, t_p$ of t (possibly $p = 0$). For each $i = 1, 2, \ldots, p$, recall that $\sigma^\star(t_i)$ is a clique, and hence there exists some node s_i of (T_t, β_t) whose bag contains the whole set $\sigma^\star(t_i)$. We construct (T'_t, β'_t) from (T_t, β_t) by attaching, for every $i = 1, \ldots, p$, the decomposition (T'_{t_i}, β'_{t_i}) as a subtree below node s_i. Since $\sigma^\star(t_i)$ is exactly the intersection of $\beta_t(s_i)$ and the root bag of (T'_{t_i}, β'_{t_i}), it can be easily verified that (T'_t, β'_t) constructed in this manner is a tree decomposition of $G[\gamma^\star(t)]$. Moreover, since $\sigma^\star(t) \subseteq \beta_t(r_t)$, then the invariant that $\sigma^\star(t)$ is contained in the root bag of (T'_t, β'_t) is preserved.

The bound on the width of (T, β) follows from the bound on the widths of decompositions (T_t, β_t) given by Theorem 3.4. For the adhesion width, observe that the only adhesions that were not present in some decomposition (T_t, β_t) are the adhesions created when attaching some (T'_{t_i}, β'_{t_i}) below the node s_i. However, these adhesions are exactly adhesions of decomposition $(T^\star, \beta^\star)$, which are of size at most $k < \rho$. Finally, observe that each bag of (T, β) originates in decomposition (T_t, β_t) for some $t \in V(T^\star)$; since (T_t, β_t) was captured by $\mathcal{B}_t$, it follows that (T, β) is captured by $\mathcal{B}$. □

5 Reducing bag sizes

Our goal in this section is to reduce the size of the bags of Theorem 4.2 from exponential to polynomial in k. This step is not needed to show fixed-parameter tractability of the isomorphism test in graphs of bounded treewidth, but reduces the running time of the algorithm from double- to single-exponential in k.

Definition 5.1. Let G be a graph, let $\mathcal{B} \subseteq 2^{V(G)}$ be a family of candidate bags, and let q be a

positive integer. Then

$$\mathcal{B}^{\leq q} := \{X \subseteq V(G) \ : \ |X| \leq q \text{ and } \exists_{B \in \mathcal{B}} X \subseteq B\}.$$

Note that if family $\mathcal{B}$ is isomorphism-invariant, then so does $\mathcal{B}^{\leq q}$.

The following lemma will be the crucial technical insight of this section. Intuitively it states that by focusing on the family $\mathcal{B}^{\leq q}$ for large enough q, instead of the original $\mathcal{B}$, we still capture some tree decomposition of the graph that has a reasonably small width. The crucial point here is that the candidate bags of $\mathcal{B}^{\leq q}$ are much smaller than those of $\mathcal{B}$. Since the canonization algorithm of Section 6 is essentially considering all permutations of all the bags, reducing the bag size will be useful for speeding it up.

Lemma 5.2. *Let G be a connected graph of treewidth less than k, and let $\mathcal{B} \subseteq 2^{V(G)}$ be a family of candidate bags that captures some tree decomposition of G that has width at most k' and adhesion width at most ℓ, where $k \leq \ell \leq k'$. Then the family $\mathcal{B}^{\leq (k+1)\ell}$ captures some tree decomposition of G that has width at most $(k+1)\ell - 1$.*

Before we proceed with the proof of Lemma 5.2, we need one more definition.

Definition 5.3 (connectivity-sensitive tree decomposition). We say that a tree decomposition (T, β) of a connected graph G is *connectivity-sensitive* (*cs-tree decomposition*, for short), if the following conditions are satisfied for every $t \in V(T)$:

- $G[\alpha(t)]$ is connected, and

- $\sigma(t) = N_G(\alpha(t))$.

Actually, one can see that the tree decomposition constructed in the proof of Theorem 3.4 is connectivity sensitive, so the family $\mathcal{B}$ actually captures a cs-tree decomposition of the graph with required width and adhesion width. A similar conclusion, however, is not so clear in the case of Theorem 4.2. Fortunately, it is easy to see that every tree decomposition of a connected graph can be turned into a cs-tree decomposition without increasing the widths. For the sake of completeness, we attach the easy proof in the appendix.

Lemma 5.4 (♠). *If a connected graph G admits a tree decomposition (T, β) of width k and adhesion width ℓ, then G admits also a cs-tree decomposition (T', β') of width at most k and adhesion width at most ℓ. Moreover, every bag appearing in (T', β') is a subset of some bag of (T, β).*

We are ready to proceed to the proof of Lemma 5.2.

Proof of Lemma 5.2. Let (T_0, β_0) be a tree decomposition of G that has width at most k' and adhesion width at most ℓ, and is captured by $\mathcal{B}$. Let (T, β) be the cs-tree decomposition of G given by Lemma 5.4 applied to (T_0, β_0). That is, (T, β) is connectivity-sensitive, has width at most k' and adhesion width at most ℓ, and every bag is contained in some bag of (T_0, β_0), so in particular in some bag of $\mathcal{B}$. We now prove that there exists a tree decomposition (T', β') of G such that:

- (T', β') has width at most $(k+1)\ell - 1$,

- every bag of (T', β') is a subset of some bag of (T, β).

From these properties it follows that (T', β') is captured by $\mathcal{B}^{\leq (k+1)\ell}$, which will conclude the proof.

We construct the decomposition (T', β') by a bottom-up induction on the decomposition (T, β). For each node $t \in V(T)$, we construct a decomposition (T'_t, β'_t) that has the following properties:

(i) (T'_t, β'_t) is a tree decomposition of $G[\gamma(t)]$ of width at most $(k+1)\ell - 1$;

(ii) every bag of (T'_t, β'_t) is a subset of some bag of (T, β);

(iii) the root bag of (T'_t, β'_t) contains $\sigma(t)$.

The decomposition (T', β') will be then simply (T'_r, β'_r), where r is the root node of (T, β).

Take any node t, and let $t_1, t_2, \ldots, t_p$ be its children in T (possibly $p = 0$ if t is a leaf). By induction hypothesis we have decompositions $\{(T'_{t_i}, \beta'_{t_i})\}_{1 \le i \le p}$ for the subtrees below t that satisfy properties (i), (ii), (iii).

Let us construct a graph H_t from $G[\gamma(t)]$ by contracting, for every $i \in \{1, 2, \ldots, p\}$, the subgraph $G[\alpha(t_i)]$ into a single vertex u_i; recall that this subgraph is connected since (T, β) is connectivity-sensitive. Moreover, by connectivity-sensitivity we have that $N_{H_t}(u_i) = \sigma(t_i)$. Since H_t is a minor of G, we infer that $\mathbf{tw}(H_t) \le \mathbf{tw}(G) < k$. Let then (T_{H_t}, β_{H_t}) be a tree decomposition of H_t of width less than k.

We construct a decomposition (T'_t, β'_t) from (T_{H_t}, β_{H_t}) in the following steps:

1. Include all the vertices of $\sigma(t)$ into each bag of (T_{H_t}, β_{H_t}).

2. Replace every occurrence of each vertex u_i in each bag by all the vertices of $N_{H_t}(u_i) = \sigma(t_i)$.

3. For every $i = 1, 2, \ldots, p$, find any node of the decomposition whose bag originally contained u_i (and so now it contains $N_{H_t}(u_i) = \sigma(t_i)$). Attach the decomposition (T'_{t_i}, β'_{t_i}) as a subtree below this node.

It is straightforward to verify that (T'_t, β'_t) constructed in this manner is a valid tree decomposition of $G[\gamma(t)]$. Moreover, observe that the bags of (T'_t, β'_t) are of size at most $(k+1)\ell$: For bags originating in decompositions (T'_{t_i}, β'_{t_i}) this follows from induction hypothesis, while bags originating in (T_{H_t}, β_{H_t}) had size at most k in the beginning, then got augmented by at most ℓ vertices in step (1), and finally some of the original vertices got replaced by ℓ other vertices in step (2). This implies that these bags have size at most $k\ell + \ell = (k+1)\ell$ at the end. This proves property (i). Properties (ii) and (iii) follow directly from the construction: Every bag originating in (T_{H_t}, β_{H_t}) is a subset of $\beta(t)$ and a superset of $\sigma(t)$, and property (ii) for bags originating in decompositions (T'_{t_i}, β'_{t_i}) follows from the induction hypothesis. $\qquad\square$

Using Lemma 5.2, we can further refine Theorem 4.2. The new property (iv) is a technical condition that will be used later.

Theorem 5.5. *Let k be a positive integer, and let G be a graph on n vertices that is connected and k-complemented. There exists an algorithm that computes an isomorphism-invariant family of bags $\mathcal{B}$ with the following properties:*

(i) $|B| \le (k+1)\rho \in \mathcal{O}(k^4)$ *for each $B \in \mathcal{B}$;*

(ii) $|\mathcal{B}| \le 2^{\mathcal{O}(k^5 \log k)} \cdot n^2$;

(iii) *assuming that $\mathbf{tw}(G) < k$, the family $\mathcal{B}$ captures some tree decomposition of G that has width at most $(k+1)\rho - 1 \in \mathcal{O}(k^4)$;*

(iv) *family $\mathcal{B}$ is closed under taking subsets.*

Moreover, the algorithm runs in $2^{\mathcal{O}(k^5 \log k)} \cdot n^3$ time.

17

Proof. We run the algorithm of Theorem 4.2 on the graph G to obtain an isomorphism-invariant family $\mathcal{B}_0$. Then, we output the family $\mathcal{B} := \mathcal{B}_0^{\leq (k+1)\rho}$. Observe that since $|B| \leq \zeta$ for each $B \in \mathcal{B}_0$, then each $B \in \mathcal{B}_0$ gives rise to at most $\sum_{i=0}^{(k+1)\rho} \binom{\zeta}{i} \in 2^{\mathcal{O}(k^5 \log k)}$ sets in the output family $\mathcal{B}$. This justifies the bound on $|\mathcal{B}|$ (property (ii)) and on the running time. Properties (i) and (iv) follow directly from the construction, and property (iii) follows from Lemma 5.2. □

6 Canonization

In this section we utilize the isomorphism-invariant family of candidate bags constructed in Theorems 3.4, 4.2, and 5.5 to give a canonization algorithm for graphs of bounded treewidth running in FPT time. Our main goal is to prove Theorem 1.3; we then deduce Theorems 1.1 and 1.2 in Section 6.3.

We proceed in two steps. First, we prove the following weaker version.

Theorem 6.1. *There exists an algorithm that, given a graph G and a positive integer k, in time $2^{\mathcal{O}(k^5 \log k)} \cdot n^5$ either correctly concludes that $\mathbf{tw}(G) \geq k$, or outputs an isomorphism-invariant term $\mathbf{t}$ that constructs G and uses at most $\mathcal{O}(k^4)$ labels. Moreover, this term has bit-length at most $\mathcal{O}(k^4 \log k \cdot n)$.*

Given Theorem 6.1, Theorem 1.3 follows from bootstraping it with the algorithm of Bodlaender [6], as shown formally in Section 6.3.

The overall structure of the algorithm of Theorem 6.1 closely follows the algorithm of Bodlaender [5]: in one subtask of a dynamic programming algorithm, one considers a pair (B, Z) where $B \subseteq V(G)$ is a candidate for a bag and Z is a vertex set of one of the connected components of $G \setminus B$, with the intention to make a tree decomposition $G[B \cup Z]$ with B as a root bag. However, instead of using all sets of bounded size as bag candidates B, we iterate over the family $\mathcal{B}$ of Theorem 5.5.

First we introduce a concept that we call *construction terms*, which is an alternative definition of treewidth and tree decompositions via graph grammars. Our canonization algorithm then produces a canonical expression (construction term) that builds the graph; the isomorphism test boils down to verifying equality of these canonical expressions.

6.1 Construction terms

The formalization given in this section has been known in the graph grammar literature from eighty's. We refer to [7, 37] for a review on these topics. We would also like to mention that the materials presented in this subsection is not much more than a formalization of what is commonly known as a nice tree decomposition [25], that is, a preprocessed decomposition, where T has bounded degree and every neighboring bags differ in at most one vertex, well-suited for describing dynamic programming algorithms on tree decompositions. We give the details here to make the presentation self-content. In the sequel, for a positive integer q we denote by $[q] = \{1, 2, \ldots, q\}$. For a function f by $f[x \to y]$ we denote a function defined as follows:

$$f[x \to y](z) = \begin{cases} y & \text{if } z = x, \\ f(z) & \text{otherwise.} \end{cases}$$

Note that this definition is correct regardless whether x is in the domain of f or not. If x belongs to the domain of f, then by $f[x \to \bot]$ we denote the function $f \setminus \{(x, f(x))\}$, i.e., f with x deleted from the domain.

Let k be a positive integer and let $\Sigma = \{1, 2, \ldots, k\}$ be an alphabet of k labels. We now define a family $\mathbb{T}$ of terms; each term $\mathbf{t} \in \mathbb{T}$ will have a prescribed subset $\mathbf{used}(\mathbf{t}) \subseteq \Sigma$ of labels *used* by $\mathbf{t}$.

- We have a *leaf term* $\mathfrak{l} \in \mathbb{T}$, with $\mathbf{used}(\mathfrak{l}) = \emptyset$.

- If $\mathbf{t} \in \mathbb{T}$ and $i \in \Sigma \setminus \mathbf{used}(\mathbf{t})$, then we can create an *introduce term* $\mathfrak{i}_i(\mathbf{t}) \in \mathbb{T}$, with $\mathbf{used}(\mathfrak{i}_i(\mathbf{t})) = \mathbf{used}(\mathbf{t}) \cup \{i\}$.

- If $\mathbf{t} \in \mathbb{T}$, $i \in \mathbf{used}(\mathbf{t})$, and $M \subseteq \mathbf{used}(\mathbf{t}) \setminus \{i\}$, then we can create a *forget term* $\mathfrak{f}_{i,M}(\mathbf{t}) \in \mathbb{T}$, with $\mathbf{used}(\mathfrak{f}_{i,M}(\mathbf{t})) = \mathbf{used}(\mathbf{t}) \setminus \{i\}$. The set M is meant to represent the neighbors of i in the graph constructed by the term.

- Let $q \geq 2$ be any integer. Suppose that there are terms $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_q \in \mathbb{T}$ such that $\mathbf{used}(\mathbf{t}_1) = \mathbf{used}(\mathbf{t}_2) = \ldots = \mathbf{used}(\mathbf{t}_q)$. Then we can create a *join term* $\mathfrak{j}(\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_q) \in \mathbb{T}$, with

$$\mathbf{used}(\mathfrak{j}(\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_q)) = \mathbf{used}(\mathbf{t}_1) = \mathbf{used}(\mathbf{t}_2) = \ldots = \mathbf{used}(\mathbf{t}_q).$$

The family $\mathbb{T}$ comprises all the terms that can be built from leaf terms using introduce, forget, introduce edge, and join terms. Note that the join terms can have arbitrarily large arity, but this arity has to be at least 2. We define the *length* of the term $\mathbf{t}$, denoted $|\mathbf{t}|$, as the total number of operators $\mathfrak{l}, \mathfrak{i}_i, \mathfrak{j}$ used in it, plus $(k' + 1)$ times the number of operations $\mathfrak{f}_{i,M}$ used in it (to account for k' bits used to encode the set M).

Terms from $\mathbb{T}$ have a natural interpretation as expressions building graphs with at most k distinguished vertices. More formally, with every term $\mathbf{t} \in \mathbb{T}$ we associate a pair $\mathfrak{G}[\mathbf{t}] = (G[\mathbf{t}], \lambda[\mathbf{t}])$, called a *labelled graph*, where $G[\mathbf{t}]$ is a graph and $\lambda[\mathbf{t}]$ is bijection between some subset of $V(G[\mathbf{t}])$ of cardinality $|\mathbf{used}(\mathbf{t})|$, and $\mathbf{used}(\mathbf{t})$. The bijection $\lambda[\mathbf{t}]$ is also called the *labelling*. We maintain the invariant that the domain of $\lambda[\mathbf{t}]$ induces an edgeless graph in $G[\mathbf{t}]$. The labelled graph $\mathfrak{G}[\cdot]$ is defined as follows:

- If $\mathbf{t} = \mathfrak{l}$, then $G[\mathfrak{l}]$ is an empty graph and $\lambda[\mathfrak{l}]$ is an empty function.

- If $\mathbf{t} = \mathfrak{i}_i(\mathbf{t}')$ for some $\mathbf{t}' \in \mathbb{T}$ and $i \in \Sigma$, then $G[\mathbf{t}]$ is equal to $G[\mathbf{t}']$ with a new independent vertex v introduced, and $\lambda[\mathbf{t}] = \lambda[\mathbf{t}'][v \to i]$.

- If $\mathbf{t} = \mathfrak{f}_{i,M}(\mathbf{t}')$ for some $\mathbf{t}' \in \mathbb{T}$, $i \in \Sigma$, and $M \subseteq \Sigma \setminus \{i\}$, then $G[\mathbf{t}]$ equals $G[\mathbf{t}']$ with additional edge $\lambda[\mathbf{t}']^{-1}(i)\lambda[\mathbf{t}']^{-1}(j)$ for every $j \in M$, and $\lambda[\mathbf{t}] = \lambda[\mathbf{t}'][\lambda[\mathbf{t}']^{-1}(i) \to \bot]$.

- Suppose $\mathbf{t} = \mathfrak{j}(\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_q)$ for some $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_q \in \mathbb{T}$. Then $G[\mathbf{t}]$ is constructed by taking the disjoint union of $G[\mathbf{t}_1], G[\mathbf{t}_2], \ldots, G[\mathbf{t}_q]$, and, for every $i \in \mathbf{used}(\mathbf{t}_1) = \mathbf{used}(\mathbf{t}_2) = \ldots = \mathbf{used}(\mathbf{t}_q)$, identifying all vertices $\{\lambda[\mathbf{t}_j]^{-1}(i) : j = 1, 2, \ldots, q\}$ into one vertex. This identified vertex is assigned label i in the labelling $\lambda[\mathbf{t}]$.

We now say that $\mathbf{t}$ is a *construction term* for graph G if $\mathbf{used}(\mathbf{t}) = \emptyset$ and $G[\mathbf{t}]$ is isomorphic to G. As the reader probably suspects, construction terms and tree decompositions are tightly related.

Lemma 6.2 (♠). *A graph G has treewidth less than k if and only if it admits a construction term that constructs it and uses at most k labels.*

Let $\mathbb{O} = \{\mathfrak{i}_i : i \in \Sigma\} \cup \{\mathfrak{f}_{i,M} : i \in \Sigma, M \subseteq \Sigma \setminus \{i\}\} \cup \{\mathfrak{l}, \mathfrak{j}\}$ be the set of operators used in the terms of $\mathbb{T}$. Let us introduce an arbitrary linear order $\trianglelefteq$ on the elements of $\mathbb{O}$: for instance first come operators $\mathfrak{i}_i$, sorted by i, then operators $\mathfrak{f}_{i,M}$, sorted lexicographically by (i, M), and finally operators $\mathfrak{l}$ and $\mathfrak{j}$. Given this order, we may inductively define a linear order $\trianglelefteq$ on the terms from $\mathbb{T}$ as follows. Let $\mathbf{t}_1, \mathbf{t}_2$ be two terms, and let $o_1, o_2 \in \mathbb{O}$ be the top-most operations used in $\mathbf{t}_1$ and $\mathbf{t}_2$, respectively. Then relation $\trianglelefteq$ between $\mathbf{t}_1$ and $\mathbf{t}_2$ is defined inductively based on the definition for terms of smaller depth.

- If $o_1 \neq o_2$, then $\mathbf{t}_1 \lhd \mathbf{t}_2$ if $o_1 \lhd o_2$, and $\mathbf{t}_1 \rhd \mathbf{t}_2$ if $o_1 \rhd o_2$.

- If $o_1 = o_2 = \mathsf{l}$, then $\mathbf{t}_1 = \mathbf{t}_2$.

- If $o_1 = o_2 \notin \{\mathsf{l}, \mathsf{j}\}$, then let $\mathbf{t}_1 = o(\mathbf{t}_1')$ and $\mathbf{t}_2 = o(\mathbf{t}_2')$, where $o = o_1 = o_2$. If $\mathbf{t}_1' = \mathbf{t}_2'$ then $\mathbf{t}_1 = \mathbf{t}_2$, if $\mathbf{t}_1' \lhd \mathbf{t}_2'$ then $\mathbf{t}_1 \lhd \mathbf{t}_2$, and if $\mathbf{t}_1' \rhd \mathbf{t}_2'$ then $\mathbf{t}_1 \rhd \mathbf{t}_2$.

- Suppose $o_1 = o_2 = \mathsf{j}$, and let the arity of the join operation in $\mathbf{t}_1, \mathbf{t}_2$ be equal to q_1, q_2, respectively. Let $\mathbf{t}_1 = \mathsf{j}(\mathbf{t}_{1,1}, \mathbf{t}_{1,2}, \ldots, \mathbf{t}_{1,q_1})$ and $\mathbf{t}_2 = \mathsf{j}(\mathbf{t}_{2,1}, \mathbf{t}_{2,2}, \ldots, \mathbf{t}_{2,q_2})$. Since terms $\mathbf{t}_{1,j}$ and $\mathbf{t}_{2,j}$ has smaller depth than $\mathbf{t}_1$ and $\mathbf{t}_2$, respectively, the order $\trianglelefteq$ is already defined for them. Hence, we may compare sequences $(\mathbf{t}_{1,1}, \mathbf{t}_{1,2}, \ldots, \mathbf{t}_{1,q_1})$ and $(\mathbf{t}_{2,1}, \mathbf{t}_{2,2}, \ldots, \mathbf{t}_{2,q_2})$ lexicographically. If $(\mathbf{t}_{1,1}, \mathbf{t}_{1,2}, \ldots, \mathbf{t}_{1,q_1}) \lhd (\mathbf{t}_{2,1}, \mathbf{t}_{2,2}, \ldots, \mathbf{t}_{2,q_2})$ then $\mathbf{t}_1 \lhd \mathbf{t}_2$, if $(\mathbf{t}_{1,1}, \mathbf{t}_{1,2}, \ldots, \mathbf{t}_{1,q_1}) \rhd (\mathbf{t}_{2,1}, \mathbf{t}_{2,2}, \ldots, \mathbf{t}_{2,q_2})$ then $\mathbf{t}_1 \rhd \mathbf{t}_2$, and if $(\mathbf{t}_{1,1}, \mathbf{t}_{1,2}, \ldots, \mathbf{t}_{1,q_1}) = (\mathbf{t}_{2,1}, \mathbf{t}_{2,2}, \ldots, \mathbf{t}_{2,q_2})$ then $\mathbf{t}_1 = \mathbf{t}_2$.

Note here that two join terms that differ only in the order of arguments are considered different, even though they construct the same labelled graph. The term where the arguments are sorted nondecreasingly is considered the smallest.

6.2 Constructing a canonical construction term

We are finally ready to prove Theorem 6.1

Proof of Theorem 6.1. Let $k' = (k+1)\rho \in \mathcal{O}(k^4)$; we will use at most k' labels. Firstly, without loss of generality we assume that G is connected. For disconnected graphs we can apply the algorithm to each connected component $G_1, G_2, \ldots, G_p$ separately, obtaining terms $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_p$, then sort these terms nondecreasingly so that $\mathbf{t}_1 \trianglelefteq \mathbf{t}_2 \trianglelefteq \ldots \trianglelefteq \mathbf{t}_p$, and output the term $\mathbf{t} := \mathsf{j}(\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_p)$. Thus, providing that the construction for a connected graph is isomorphism-invariant, then due to the sorting step so is the construction for disconnected graphs.

We now compute the k-improved graph $G^{\langle k \rangle}$, using Lemma 2.8. If computation of this graph revealed that $\mathbf{tw}(G) \geq k$, then we provide a negative answer to the whole algorithm. Now, we apply Theorem 5.5 to the graph $G^{\langle k \rangle}$, obtaining an isomorphism-invariant family of candidate bags $\mathcal{B}$. Observe that since the definition of $G^{\langle k \rangle}$ is invariant w.r.t. isomorphisms of G, and the definition of $\mathcal{B}$ is invariant w.r.t. isomorphisms of $G^{\langle k \rangle}$, then the family $\mathcal{B}$ is invariant w.r.t. isomorphisms of G.

Assume for a moment that G has a tree decomposition of width less than k. Then, by Lemma 2.7, so does $G^{\langle k \rangle}$. Consequently, by Theorem 5.5 we have that $\mathcal{B}$ captures some tree decomposition of $G^{\langle k \rangle}$ that has width at most $k' - 1$. Since $G^{\langle k \rangle}$ is a supergraph of G, this tree decomposition is also a tree decomposition of G. By Lemma 5.4 and property (iv) of Theorem 5.5, we can further infer that $\mathcal{B}$ captures some cs-tree decomposition of G of width at most $k' - 1$. Let us denote this cs-tree decomposition by (T, β).

The plan for the rest of the proof is as follows. We provide a dynamic programming algorithm that exploits the family $\mathcal{B}$ to compute a term $\mathbf{t}$ that constructs G. From the algorithm it will be clear that the definition of $\mathbf{t}$ is isomorphism-invariant. It is possible that the computation of $\mathbf{t}$ fails, but only if $\mathbf{tw}(G) \geq k$: Using the captured cs-tree decomposition (T, β), we will argue that the algorithm computes some feasible construction term, providing that $\mathbf{tw}(G) < k$. Hence, in case of failure we can safely report that $\mathbf{tw}(G) \geq k$.

Let us define the family of *states* $\mathbb{S}$ as the family of all the triples (B, λ, Z), where

- $B \in \mathcal{B}$;

- λ is an injective function from B to $[k']$;

- $Z = \emptyset$ or Z is the vertex set of a connected component of $G \setminus B$.

Observe that $|\mathbb{S}| \leq |\mathcal{B}| \cdot k'! \cdot (n+1) = 2^{\mathcal{O}(k^5 \log k)} \cdot n^3$. For every state $I = (B, \lambda, Z) \in \mathbb{S}$, we compute a term $\mathbf{t}[I]$ that constructs the labeled graph $\mathfrak{G}[I] := (G[B \cup Z] \setminus \binom{B}{2}, \lambda)$, i.e., the graph $G[B \cup Z]$ with all the edges inside B cleared, and with labelling λ on B. The definition of $\mathbf{t}[I]$ will be invariant w.r.t. isomorphisms of the structure $\mathfrak{G}[I]$. Computation of $\mathbf{t}[I]$ can possibly fail, in which case we denote it by $\mathbf{t}[I] = \bot$. The output term $\mathbf{t}$ is simply defined as $\mathbf{t}[\emptyset, \emptyset, V(G)]$, and using the captured cs-tree decomposition (T, β) we will make sure that $\mathbf{t}[\emptyset, \emptyset, V(G)] \neq \bot$ in case $\mathbf{tw}(G) < k$.

To make sure that the inductive definition of $\mathbf{t}[I]$ is well-defined, we also define the *potential* Φ of a state $I = (B, \lambda, Z)$ similar to the one defined in Theorem 3.4: $\Phi(B, \lambda, Z) = 2|Z| + |B|$. The definition of $\mathbf{t}[I]$ depends only on the terms for states with a strictly smaller potential. Since the potential is always nonnegative, the definition is valid.

Before we proceed to the definition of $\mathbf{t}[I]$, let us introduce one more helpful definition. We often run into situations where we would like to compute the canonical term for a triple (B, λ, Z) that is not necessarily a state according to our definition, because Z consists of several connected components of $G \setminus B$ rather than at most one. To cope with such situations, we define operator $\mathtt{break}[B, \lambda, Z]$. Formally, operator $\mathtt{break}[B, \lambda, Z]$ can be applied to triples (B, λ, Z) where $B \in \mathcal{B}$, λ is an injective function from B to $[k']$, and Z comprises vertex sets of some (possibly zero) connected components of $G \setminus B$. The behaviour of $\mathtt{break}[B, \lambda, Z]$ is defined as follows:

- If $Z = \emptyset$ or $G[Z]$ is connected (equivalently, $(B, \lambda, Z) \in \mathbb{S}$), then we simply put $\mathtt{break}[B, \lambda, Z] = \mathbf{t}[B, \lambda, Z]$.

- If $G[Z]$ consists of more than one connected component, then let $Z_1, Z_2, \ldots, Z_p$ be the vertex sets of these connected components. Let $\mathbf{t}_i = \mathbf{t}[B, \lambda, Z_i]$ for $i = 1, 2, \ldots, p$. If any of the terms $\mathbf{t}_i$ is equal to $\bot$, then we put $\mathtt{break}[B, \lambda, Z] = \bot$. Otherwise, by sorting the terms if necessary, assume that $\mathbf{t}_1 \trianglelefteq \mathbf{t}_2 \trianglelefteq \ldots \trianglelefteq \mathbf{t}_p$. Then $\mathtt{break}[B, \lambda, Z] = \mathsf{j}(\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_p)$.

Observe that the join operation is valid, since we assumed that term $\mathbf{t}_i$ constructs $(G[B \cup Z_i] \setminus \binom{B}{2}, \lambda)$, where all the edges between the vertices of B are cleared. We naturally extend the notation $\mathfrak{G}[\cdot]$ to triples that can be arguments of the operator $\mathtt{break}[\cdot]$.

We now proceed to the definition of $\mathbf{t}[I]$ for a state $I = (B, \lambda, Z) \in \mathbb{S}$. We generate a family $\mathcal{C}$ of candidates for $\mathbf{t}[I]$. We put $\mathbf{t}[I] = \bot$ if $\mathcal{C} = \emptyset$, and otherwise $\mathbf{t}[I]$ is defined as the $\trianglelefteq$-minimum element of $\mathcal{C}$. Elements of $\mathcal{C}$ reflect possible ways of obtaining the term constructing $\mathfrak{G}[I]$ from simpler terms.

Firstly, if $Z = B = \emptyset$, then we take $\mathcal{C} = \{\mathfrak{l}\}$.

Assume now that B contains some vertex u that is not adjacent to any vertex of Z. Then, for every such vertex u, we add to $\mathcal{C}$ the term $\mathbf{t}_{\mathsf{i},u} := \mathsf{i}_{\lambda(u)}(\mathbf{t}[I'])$ for $I' = (B \setminus u, \lambda[u \to \bot], Z)$. Formally, we add this term only if $I' \in \mathbb{S}$ and $\mathbf{t}[I] \neq \bot$; the same remark holds also for the other elements of $\mathcal{C}$ to follow. Observe that if $\mathbf{t}[I']$ constructs $\mathfrak{G}[I']$, then $\mathbf{t}_{\mathsf{i},u}$ constructs $\mathfrak{G}[I]$.

Then, for every vertex $v \in Z$ we consider the possibility that v has just been forgotten. Formally, for each $v \in Z$ and each label $i \in [k'] \setminus \lambda(B)$, we add to $\mathcal{C}$ the following term:

$$\mathbf{t}_{\mathsf{f},v,i} := \mathsf{f}_{i,M}(\mathtt{break}[I']), \tag{3}$$

where $I' = (B \cup \{v\}, \lambda[v \to i], Z \setminus v)$ and M is the set of labels in λ of neighbors of v in B in the graph G. Again, if $\mathtt{break}[\cdot]$ cannot be applied to I' or if $\mathtt{break}[I'] = \bot$, then we do not add this candidate. Observe that if $\mathtt{break}[I']$ constructs $\mathfrak{G}[I']$, then $\mathbf{t}_{\mathsf{f},v,i}$ constructs $\mathfrak{G}[I]$.

This concludes the definition of the term $\mathbf{t}[I]$; observe that the definition depends only on the definitions for states with strictly smaller potential, as was promised. It can be easily seen by induction that the definition is invariant with respect to isomorphisms of the structure $(G, \mathfrak{G}[I])$,

due to taking the $\trianglelefteq$-minimum from an invariant family of candidates. The following claim shows that, in the end, we obtain a meaningful term provided that $\mathbf{tw}(G) < k$.

Claim 6.3. *If* $\mathbf{tw}(G) < k$ *then* $\mathbf{t}[\emptyset, \emptyset, V(G)] \neq \bot$.

Proof. We proceed by a bottom-up induction on the decomposition (T, β). For any $t \in V(T)$ and any injective labelling $\lambda \colon \sigma(t) \to [k']$, define $I_{t,\lambda} := (\sigma(t), \lambda, \alpha(t))$. Observe that $I_{t,\lambda} \in \mathbb{S}$, since $\sigma(t) \subseteq \beta(t) \in \mathcal{B}$ and $\mathcal{B}$ is closed under taking subsets, and $G[\alpha(t)]$ is connected since (T, β) is connectivity-sensitive. We prove inductively the following statement:

> For each $t \in V(T)$ and any injective labelling $\lambda \colon \sigma(t) \to [k']$, we have $\mathbf{t}[I_{t,\lambda}] \neq \bot$. $\quad$ (4)

Observe that if r is the root of T, then $I_{r,\emptyset} = (\emptyset, \emptyset, V(G))$, so the statement (4) for r is equivalent to the statement of the claim.

Let $t_1, t_2, \ldots, t_p$ be the children of t in T (possibly $p = 0$). It is more convenient to prove an even stronger statement:

> For any X such that $\sigma(t) \subseteq X \subseteq \beta(t)$, any labeling $\lambda_X \colon X \to [k']$ that extends λ,
>
> and any Z that is the vertex set of some connected component of $G[\gamma(t)] \setminus X$, $\quad$ (5)
>
> it holds that $\mathbf{t}[X, \lambda_X, Z] \neq \bot$.

Observe that, again, $(X, \lambda_X, Z) \in \mathbb{S}$ since $X \subseteq \beta(t)$. Statement (4) for t, which we are trying to prove, is equivalent to statement (5) for $X = \sigma(t)$, $\lambda_X = \lambda$ and $Z = \alpha(t)$. We prove statement (5) for all choices of X, λ_X, Z by induction with respect to $|Z \cap \beta(t)|$.

For the base of the induction, observe that if $Z \cap \beta(t) = \emptyset$, then $Z = \alpha(t_i)$ for some $i \in \{1, 2, \ldots, p\}$, since (T, β) is connectivity-sensitive. Moreover, since Z is a connected component of $G[\gamma(t)] \setminus X$, then $X \supseteq N(Z) = \sigma(t_i)$. By induction hypothesis for statement (4), we have that $\mathbf{t}[\sigma(t_i), \lambda_X|_{\sigma(t_i)}, \alpha(t_i)] \neq \bot$. Then it follows that the term

$$\mathbf{i}_{u_1, \lambda_X(u_1)}(\mathbf{i}_{u_2, \lambda_X(u_2)}(\ldots \mathbf{i}_{u_c, \lambda_X(u_c)}(\mathbf{t}[\sigma(t_i), \lambda_X|_{\sigma(t_i)}, \alpha(t_i)]) \ldots))$$

is among the candidates for $\mathbf{t}[X, \lambda_X, Z]$, where $(u_1, u_2, \ldots, u_c)$ is an arbitrary enumeration of $X \setminus \sigma(t_i)$. This proves that $\mathbf{t}[X, \lambda_X, Z] \neq \bot$.

Consider now the induction step when $Z \cap \beta(t)$ is non-empty. Let v be any vertex of $Z \cap \beta(t)$, and let i be any label from $[k'] \setminus \lambda_X(X)$; since $X \subsetneq \beta(t)$ and $|\beta(t)| \leq k'$, such a label exists. Observe now that from the induction hypothesis for statement (5) it follows that $\mathtt{break}[X \cup \{v\}, \lambda_X[v \to i], Z \setminus \{v\}] \neq \bot$, since in the definition of $\mathtt{break}[\cdot]$ the set $Z \setminus \{v\}$ can only be partitioned into smaller connected components, each of them with a strictly smaller intersection with $\beta(t)$ than $Z \cap \beta(t)$. Therefore, the term $\mathbf{t}_{\mathsf{f},v,i}$ defined as in (3) is among the candidates for the value of $\mathbf{t}[X, \lambda_X, Z]$, which proves that $\mathbf{t}[X, \lambda_X, Z] \neq \bot$.

This concludes the inductive proof of statement (5) for all choices of X, λ_X, Z, which also proves the induction step for statement (4) (both for leaf and non-leaf nodes). As explained earlier, statement (4) for the root of the decomposition proves that the algorithm is correct, i.e., it outputs some construction term providing that $\mathbf{tw}(G) < k$. $\quad \lrcorner$

We are left with establishing the upper bound on the length of the output term, and analysing the running time of the algorithm. To achieve this goal, we inductively bound the lengths of the terms produced by the algorithm. For a state $I = (B, \lambda, Z)$, define $\phi(I)$ as follows:

$$\phi(I) = (k' + 2) \cdot \max(2|Z| - 1, 0) + |B| + 2. \quad (6)$$

22

Observe that if $|Z| \geq 1$, then

$$\phi(I) \leq (k' + 2) \cdot 2|Z|. \tag{7}$$

Claim 6.4. *For any $I \in \mathbb{S}$, if $\mathbf{t}[I] \neq \bot$ then $|\mathbf{t}[I]| \leq \phi(I)$.*

Proof. We first verify the claim for states I where $Z = \emptyset$. Then it is easy to see that $\mathbf{t}[I]$ consists of a sequence of introduce terms that introduce consecutive vertices, finished by a leaf term. Therefore $|\mathbf{t}[I]| = |B| + 1 \leq \phi(I)$. In the sequel we assume that $Z \neq \emptyset$, and we proceed by induction with respect to the potential $\Phi(I)$.

Assume now that $Z \neq \emptyset$ and that $\mathbf{t}[B, \lambda, Z] = \mathbf{t}_{\mathbf{i}, u} = \mathbf{i}_{\lambda(u)}(\mathbf{t}[I'])$ for some vertex $u \in B$, where $I' = (B \setminus u, \lambda[u \to \bot], Z)$. Then, using the induction hypothesis we have that:

$$|\mathbf{t}[I]| = 1 + |\mathbf{t}[I']| \leq 1 + \phi(I) = 1 + (k' + 2) \cdot (2|Z| - 1) + (|B| - 1) + 2 = \phi(I).$$

Finally, assume that $\mathbf{t}[B, \lambda, Z] = \mathbf{t}_{\mathbf{f}, v, i}$, where $\mathbf{t}_{\mathbf{f}, v, i}$ is defined as in (3). Then we have

$$|\mathbf{t}[B, \lambda, Z]| \leq 1 + k' + |\mathtt{break}[B \cup \{v\}, \lambda[v \to i], Z \setminus \{v\}]|.$$

Let $Z_1, Z_2, \ldots, Z_p$ be the vertex sets of the connected components of $G[Z \setminus \{v\}]$ (possibly $p = 0$), and let $I_j = (B \cup \{v\}, \lambda[v \to i], Z_j)$ for $j = 1, 2, \ldots, p$.

Firstly, we consider the case when $p = 0$, or equivalently $Z = \{v\}$. Using our observations about the case $Z = \emptyset$ we infer that $|\mathtt{break}[B \cup \{v\}, \lambda[v \to i], Z \setminus \{v\}]| = |B| + 2$. Then

$$|\mathbf{t}[B, \lambda, Z]| \leq |B| + k' + 3 \leq (k' + 2) + |B| + 1 < \phi(I).$$

Secondly, we consider the case when $p > 0$. Using inequality (7) and the fact that $|Z_j| \geq 1$ for each $j = 1, 2 \ldots, p$, we infer that

$$|\mathtt{break}[B \cup \{v\}, \lambda[v \to i], Z \setminus \{v\}]| \leq 1 + \sum_{i=1}^{p} \phi(I_p) \leq 1 + \sum_{i=1}^{p} (k' + 2) \cdot 2|Z_i| \leq 1 + (k' + 2) \cdot 2(|Z| - 1).$$

Therefore, we obtain that

$$|\mathbf{t}[B, \lambda, Z]| \leq 1 + |B| + 1 + (k' + 2) \cdot 2(|Z| - 1) \leq \phi(I),$$

which concludes the proof of the claim. ⌟

Claim 6.4 implies that each term $\mathbf{t}[I]$ computed by the algorithm has length at most $\mathcal{O}(k'n)$. Since in the definition of length we count every term $\mathfrak{f}_{i, M}$ with weight $k' + 1$, such a term can be encoded in bit-length $\mathcal{O}(k' \log k'n)$; the extra $\log k'$ factor is used to encode the index i in the $\mathbf{i}_i$ terms.

For the analysis of the running time of the algorithm, observe that for each state $I \in \mathbb{S}$ we consider $\mathcal{O}(k'n)$ possible candidates for $\mathbf{t}[I]$. Each of these candidates is constructed in $\mathcal{O}(kn)$ time, since we possibly need to partition $G[Z \setminus \{v\}]$ into connected components. Moreover, each of these candidates has length at most $\mathcal{O}(k'n)$, by Claim 6.4. It follows that the $\trianglelefteq$-minimum among these candidates can be selected in $\mathcal{O}((k'n)^2)$ time. Since $|\mathbb{S}| = 2^{\mathcal{O}(k^5 \log k)} \cdot n^3$, we conclude that the whole algorithm works in time $2^{\mathcal{O}(k^5 \log k)} \cdot n^5$. □

6.3 Corollaries of Theorem 6.1

We first deduce Theorem 1.3 from Theorem 6.1.

Proof of Theorem 1.3. Recall that the algorithm of Bodlaender [6], given a graph G and an integer k, in time $2^{\mathcal{O}(k^3)}n$ either finds a tree decomposition of G of width less than k, or concludes that $\mathbf{tw}(G) \geq k$.

Given a graph G, we compute the canonical term $\mathbf{t}$ constructing G using Theorem 6.1, and apply the algorithm of Bodlaender [6] to the graph $G[\mathbf{t}]$ and parameter k, obtaining a tree decompositon (T, β). If the computation of $\mathbf{t}$ or the algorithm of Bodlaender returned that the treewidth of G or $G[\mathbf{t}]$ is at least k, then we return the same answer. Otherwise, we encode the output decomposition (T, β) as a term exactly as it is done in the proof of Lemma 6.2, trying every possible bag as a root bag, and picking the lexicographically minimum term as the output. Since the algorithm of Bodlaender is deterministic, the output decomposition (T, β) and the finally constructed term is isomorphic-invariant. Furthermore, as the decomposition (T, β) contains $\mathcal{O}(n)$ bags, the output term is of bit-length $\mathcal{O}(k \log k \cdot n)$. $\qquad\square$

We now show how Theorems 1.1 and 1.2 follow directly from Theorem 1.3.

Proof of Theorem 1.1. We run the algorithm of Theorem 1.3 on both G_1 and G_2. If for any of them the algorithm concluded that the treewidth is at least k, then we output the appropriate answer. Otherwise, the algorithm returned two terms $\mathbf{t}_1, \mathbf{t}_2$ that construct G_1, G_2, respectively. Since $\mathbf{t}_1, \mathbf{t}_2$ are isomorphism-invariant, to verify whether G_1 and G_2 are isomorphic it suffices to check whether $\mathbf{t}_1 = \mathbf{t}_2$. $\qquad\square$

Proof of Theorem 1.2. Given a graph G, we compute the canonical term $\mathbf{t}$ constructing G, construct an ordering $\phi : V(G) \to [n]$ of the vertices of G according to pre-order in term $\mathbf{t}$ of operations when they become forgotten, and output the graph $G[\mathbf{t}]$ on the vertex set $[n]$ together with the mapping ϕ. If the computation of $\mathbf{t}$ returned that the treewidth of G is at least k, then we return the same answer. $\qquad\square$

7 Conclusions

In this paper we have developed the first fixed-parameter tractable algorithm for GRAPH ISOMORPHISM parameterized by treewidth. The obvious open question is to improve the running time of our algorithm.

In this work we focused on keeping the presentation as clear as possible, while targeting at a $2^{\text{poly}(k)}$ FPT algorithm at the same time — but without any attempt of optimizing the $\text{poly}(k)$ factor in the exponent, nor the polynomial factor in n. Although it is reasonable to suspect that the polynomial factor in n in the running time of our algorithm can be reduced to n^4, or even n^3, by a more careful analysis, we consider such an improvement of minor importance, and the more challenging question would be to make the whole algorithm run in quadratic, or even linear time. Recall that isomorphism of trees can be verified in linear time [1], so there is no reason why such a running time should not be achieved also for graphs of bounded treewidth.

A possible route to improving the polynomial factor could be the alternative approach proposed by Otachi and Schweitzer [31]. In essence, Otachi and Schweitzer show that once an isomorphism-invariant family of potential bags of size $f(k) \cdot n^c$ is constructed, then an FPT isomorphism test can be performed using a variant of the *Weisfeiler-Lehman algorithm*, which thus can serve as an alternative to our dynamic programming procedure of Section 6. Since the Weisfeiler-Lehman

algorithm is very simple, it is possible that the combination of our enumeration algorithm and the techniques of Otachi and Schweitzer can lead to improving the polynomial factor.

For the parametric dependence, we also believe that the factor $2^{\mathcal{O}(k^5 \log k)}$ is suboptimal. A challenging question would be to improve it to $2^{\mathcal{O}(k \log k)}$ or even $2^{\mathcal{O}(k)}$. In the current approach, the most significant reason for such a high polynomial in the exponent is the way we handle small sets S in the proof of Lemma 3.3.

It is also interesting to investigate whether the results of our work can be used to prove canonical or almost canonical variants of other graph decompositions. Actually, many structural theorems for graphs follow the general approach proposed by Robertson and Seymour in their approximation algorithm for treewidth [33]. In particular, the step of breaking the top adhesion S using a small separator has been used multiple times in various works. Since our work provides a canonical way of performing this step (encompassed in Lemmas 3.2 and Lemma 3.3), it might serve as a solid foundation for making other graph decompositions canonical. For concrete decomposition theorems where we hope that our approach could be applicable, let us name (a) the H-minor-free structural theorem of Robertson and Seymour [34], (b) the H-topological-minor-free structural theorem of Grohe and Marx [18], and (c) the decomposition theorem with unbreakable parts given by a superset of the current authors [13].

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

[2] S. Arnborg and A. Proskurowski. Canonical representations of partial 2- and 3-trees. *BIT*, 32(2):197–214, 1992.

[3] L. Babai and E. M. Luks. Canonical labeling of graphs. In *STOC*, pages 171–183, 1983.

[4] A. Berry, R. Pogorelcnik, and G. Simonet. An introduction to clique minimal separator decomposition. *Algorithms*, 3(2):197–215, 2010.

[5] H. L. Bodlaender. Polynomial algorithms for Graph Isomorphism and Chromatic Index on partial k-trees. *J. Algorithms*, 11(4):631–643, 1990.

[6] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

[7] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998.

[8] H. L. Bodlaender. Necessary edges in k-chordalisations of graphs. *J. Comb. Optim.*, 7(3):283–290, 2003.

[9] H. L. Bodlaender, E. D. Demaine, M. R. Fellows, J. Guo, D. Hermelin, D. Lokshtanov, M. Müller, V. Raman, J. M. M. van Rooij, and F. A. Rosamond. Open problems in parameterized and exact computation — IWPEC 2008. *Technical Report UU-CS-2008-017, Department of Information and Computing Sciences, Utrecht University*, 2008.

[10] A. Bouland, A. Dawar, and E. Kopczyński. On tractable parameterizations of Graph Isomorphism. In *IPEC*, pages 218–230, 2012.

[11] F. Clautiaux, J. Carlier, A. Moukrim, and S. Nègre. New lower and upper bounds for graph treewidth. In *Experimental and Efficient Algorithms, Second International Workshop, WEA 2003, Ascona, Switzerland, May 26-28, 2003, Proceedings*, volume 2647 of *Lecture Notes in Computer Science*, pages 70–80. Springer, 2003.

[12] B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic — A Language-*

Theoretic Approach, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012.

[13] M. Cygan, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. Minimum bisection is fixed parameter tractable. In *STOC*, pages 323–332, 2014.

[14] R. Diestel. *Graph Theory*. Springer, 2005.

[15] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[16] I. S. Filotti and J. N. Mayer. A polynomial-time algorithm for determining the isomorphism of graphs of fixed genus (working paper). In *STOC*, pages 236–243, 1980.

[17] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2006.

[18] M. Grohe and D. Marx. Structure theorem and isomorphism test for graphs with excluded topological subgraphs. *SIAM J. Comput.*, 44(1):114–159, 2015.

[19] G. Z. Gutin, K. Iwama, and D. M. Thilikos. Parameterized complexity and the understanding, design, and analysis of heuristics. *NII Shonan Meeting Report*, 2013-2, 2013.

[20] J. E. Hopcroft and R. E. Tarjan. Isomorphism of planar graphs. In *Complexity of Computer Computations*, pages 131–152, 1972.

[21] J. E. Hopcroft and R. E. Tarjan. A $v \log v$ algorithm for isomorphism of triconnected planar graphs. *J. Comput. Syst. Sci.*, 7(3):323–331, 1973.

[22] J. E. Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs (preliminary report). In *STOC*, pages 172–184, 1974.

[23] K. Kawarabayashi and B. Mohar. Graph and map isomorphism and all polyhedral embeddings in linear time. In *STOC*, pages 471–480, 2008.

[24] J. M. Kleinberg and É. Tardos. *Algorithm design*. Addison-Wesley, 2006.

[25] T. Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.

[26] S. Kratsch and P. Schweitzer. Isomorphism for graphs of bounded feedback vertex set number. In *SWAT*, pages 81–92, 2010.

[27] H. Leimer. Optimal decomposition by clique separators. *Discrete Mathematics*, 113(1-3):99–123, 1993.

[28] E. M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *J. Comput. Syst. Sci.*, 25(1):42–65, 1982.

[29] G. L. Miller. Isomorphism testing for graphs of bounded genus. In *STOC*, pages 225–235, 1980.

[30] Y. Otachi. Isomorphism for graphs of bounded connected-path-distance-width. In *ISAAC*, pages 455–464, 2012.

[31] Y. Otachi and P. Schweitzer. Reduction techniques for Graph Isomorphism in the context of width parameters. *CoRR*, abs/1403.7238, 2014.

[32] I. Ponomarenko. The isomorphism problem for classes of graphs closed under contraction. *Journal of Soviet Mathematics*, 55(2):1621–1643, 1991.

[33] N. Robertson and P. D. Seymour. Graph Minors XIII. The Disjoint Paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995.

[34] N. Robertson and P. D. Seymour. Graph Minors XVI. Excluding a non-planar graph. *J. Comb. Theory, Ser. B*, 89(1):43–76, 2003.

[35] U. Schöning. Graph Isomorphism is in the low hierarchy. *J. Comput. Syst. Sci.*, 37(3):312–323, 1988.

[36] R. E. Tarjan. Decomposition by clique separators. *Discrete Mathematics*, 55(2):221–232, 1985.

[37] R. van Bevern, M. R. Fellows, S. Gaspers, and F. A. Rosamond. Myhill-Nerode methods for hypergraphs. In *ISAAC*, volume 8283 of *Lecture Notes in Computer Science*, pages 372–382, 2013.

[38] H. Weinberg. A simple and efficient algorithm for determining isomorphism of planar triply connected graphs. *Circuit Theory*, 13:142–148, 1966.

[39] K. Yamazaki, H. L. Bodlaender, B. de Fluiter, and D. M. Thilikos. Isomorphism for graphs of bounded distance width. *Algorithmica*, 24(2):105–127, 1999.

Appendix

Proof of Lemma 2.6. Let $H = G^{\langle k \rangle}$. Take any $x, y \in V(G)$ and assume that $xy \notin E(H)$, so in particular $xy \notin E(G)$. Then by the definition of $H = G^{\langle k \rangle}$, we have that the maximum vertex flow between x and y in G has size less than k. By Menger's theorem this means that there exists a separation (A, B) of G that has order less than k, and such that $x \in A \setminus B$ and $y \in B \setminus A$. We claim that (A, B) is also a separation of H. Indeed, for any pair of vertices $u \in A \setminus B$ and $v \in B \setminus A$, the fact that $|A \cap B| < k$ certifies that $\mu_G(u, v) < k$, which means that $uv \notin E(H)$ by the definition of H. Since $x \in A \setminus B$, $y \in B \setminus A$, and (A, B) is a separation of order less than k in H, then this certifies that $\mu_H(x, y) < k$. As x, y were chosen arbitrarily, it follows that H is k-complemented. $\square$

Proof of Lemma 2.7. Let (T, β) be a tree decomposition of G of width less than k. In order to show that (T, β) is also a tree decomposition of $H := G^{\langle k \rangle}$, it suffices to show that for any $xy \in E(H) \setminus E(G)$ there exists some $t \in T$ such that $x, y \in \beta(t)$.

For the sake of contradiction assume that no such t exists. Let T_x^0 and T_y^0 be the subtrees of T induced by the nodes whose bags contain x and y, respectively. We know that T_x^0 and T_y^0 are connected and vertex-disjoint. Let T_y be the connected component of $T \setminus V(T_x^0)$ that contains T_y^0 as a subgraph, and let $T_x = T \setminus V(T_y)$; note that T_x is connected and contains T_x^0 as a subgraph. Observe that $(V(T_x), V(T_y))$ forms a partition of $V(T)$. Let $A = \bigcup_{t \in V(T_x)} \beta(t)$ and $B = \bigcup_{t \in V(T_y)} \beta(t)$; note that $x \in A \setminus B$ and $y \in B \setminus A$. Observe that since T is a tree, there exists only one edge $t_x t_y$ of T that connects a node from $V(T_x)$ with a node from $V(T_y)$. From the properties of a tree decomposition it follows that $A \cap B = \beta(t_x) \cap \beta(t_y)$. Moreover, observe that $x \in \beta(t_x) \setminus \beta(t_y)$, thus $|A \cap B| = |\beta(t_x) \cap \beta(t_y)| < |\beta(t_x)| \le k$. As every vertex of G is contained in some bag of (T, β), it follows that (A, B) is a separation of order less than k that separates x and y. This proves that $\mu_G(x, y) < k$, contradicting the assumption that $xy \in E(H)$. $\square$

Proof of Lemma 2.8. If $|E(G)| > (k - 1)n$, then we can output that $\mathbf{tw}(G) \ge k$, since a graph of treewidth less than k is $(k-1)$-degenerate. Hence assume that $|E(G)| \le (k-1)n$. We perform a brute-force algorithm that follows immediately from the definition: For every pair of vertices, we compute the maximum flow between them using Lemma 2.2 with the bound k on the size of the cut, taking time $\mathcal{O}(k^2 n)$. Since we perform this procedure for every pair of vertices, the running time of $\mathcal{O}(k^2 n^3)$ follows. $\square$

Proof of Lemma 5.4. Let (T, β) be a tree decomposition of G of width k and adhesion width ℓ. We prove the following statement by a bottom-up induction on (T, β): For every $t \in V(T)$ and every vertex set Z of a connected component of $G[\alpha(t)]$, there exists a tree decomposition $(T_{t,Z}, \beta_{t,Z})$ of $G[N[Z]]$ such that:

(a) $(T_{t,Z}, \beta_{t,Z})$ is connectivity-sensitive,

(b) $(T_{t,Z}, \beta_{t,Z})$ has width at most k and adhesion width at most ℓ,

(c) every bag of $(T_{t,Z}, \beta_{t,Z})$ is a subset of some bag of (T, β), and

(d) $N(Z)$ is contained in the root bag of $(T_{t,Z}, \beta_{t,Z})$.

Observe that in this definition it holds that $N(Z) \subseteq \sigma(t)$. Since G is connected, for the final decomposition (T', β') we may take $(T_{r,V(G)}, \beta_{r,V(G)})$, where r is the root of (T, β).

Let us focus on one choice of t, Z. Let $t_1, t_2, \ldots, t_p$ be the children of t in T (possibly $p = 0$). For $i = 1, 2, \ldots, p$, let $Z_i = Z \cap \alpha(t_i)$, and let $Z_i^1, Z_i^2, \ldots, Z_i^{q_i}$ be the vertex sets of the connected components of $G[Z_i]$. For $i = 1, 2, \ldots, p$ and $j = 1, 2, \ldots, q_i$, let $(T_{t_i, Z_i^j}, \beta_{t_i, Z_i^j})$ be the decomposition

of $G[N[Z_i^j]]$ that satisfies properties (a), (b), (c), (d); existence of this decomposition is asserted by the induction hypothesis. We construct the decomposition $(T_{t,Z}, \beta_{t,Z})$ by creating one bag $\beta(t) \cap N[Z]$, and attaching all the decompositions $(T_{t_i,Z_i^j}, \beta_{t_i,Z_i^j})$ below it as subtrees. Let t_i^j be the root node of the attached decomposition $(T_{t_i,Z_i^j}, \beta_{t_i,Z_i^j})$. It is straightforward to verify that $(T_{t,Z}, \beta_{t,Z})$ is indeed a tree decomposition of $G[N[Z]]$. We now verify that the requested properties are satisfied.

- For property (a), the only checks not implied by the induction hypothesis are as follows:

    - We need to verify that $G[N[Z]]$ is connected, but this follows from the fact that $G[Z]$ is connected.
    - We need to verify that $G[\alpha_{t,Z}(t_i^j)]$ is connected and that $N(\alpha_{t,Z}(t_i^j)) = \sigma_{t,Z}(t_i^j)$. However, we have that $\alpha_{t,Z}(t_i^j) = Z_i^j$, which induces a connected graph by its definition, and that $\sigma_{t,Z}(t_i^j) = N(Z_i^j)$ by the definition of $(T_{t_i,Z_i^j}, \beta_{t_i,Z_i^j})$ and property (d) for it.

- For properties (b) and (c), it suffices to observe that $\beta(t) \cap N[Z] \subseteq \beta(t)$ and that $\sigma_{t,Z}(t_i^j) \subseteq \sigma(t_i)$.

- Property (d) follows directly from the definition of Z and of the top bag of $(T_{t,Z}, \beta_{t,Z})$.

This concludes the step of the induction. $\qquad\square$

Proof of Lemma 6.2. From right to left, let $\mathbf{t}$ be a term that constructs G and uses at most k labels. For each subterm $\mathbf{t}'$ of $\mathbf{t}$, create one node $t_{\mathbf{t}'}$ with associated bag $\beta(t_{\mathbf{t}'})$ being the domain of $\lambda[\mathbf{t}']$. Since $\mathbf{t}$ uses at most k labels, it follows that $|\beta(t_{\mathbf{t}'})| \le k$. Now connect nodes $t_{\mathbf{t}'}$ into a tree using the structure inherited from the term $\mathbf{t}$: For any two subterms $\mathbf{t}'$, $\mathbf{t}''$, connect $t_{\mathbf{t}'}$ and $t_{\mathbf{t}''}$ if and only if $\mathbf{t}''$ appears as an argument of the top-most operation in $\mathbf{t}'$, or vice-versa. Let T be the obtained tree. It is straightforward to verify that the (T, β) is a tree decomposition of G, and we already verified that it has width less than k.

From left to right, let (T, β) be a tree decomposition of G of width less than k. We apply a bottom-up induction on (T, β). More precisely, for every $t \in V(T)$ and every injective labeling $\lambda \colon \sigma(t) \to [k]$ we construct a term $\mathbf{t}_{t,\lambda}$ that constructs $(G[\gamma(t)] \setminus \binom{\sigma(t)}{2}, \lambda)$, i.e., the graph $G[\gamma(t)]$ with all the edges inside $\sigma(t)$ cleared, and with labeling λ on $\sigma(t)$. The final term $\mathbf{t}$ will be just $\mathbf{t}_{r,\emptyset}$, where r is the root of the tree T.

Let us take any $t \in V(T)$, and let $t_1, t_2, \ldots, t_p$ be the children of t in T (possibly $p = 0$). Let λ' be any injective extension of λ onto $\beta(t)$; such an extension exists since $|\beta(t)| \le k$. We first construct an auxiliary term $\mathbf{t}'$, which will construct $(G[\gamma(t)] \setminus \binom{\beta(t)}{2}, \lambda')$. The construction of $\mathbf{t}'$ distinguishes two cases: either $p = 0$ or $p > 0$.

Consider first the case when $p = 0$, i.e., t is a leaf node. Then we can take

$$\mathbf{t}' = \mathsf{i}_{\lambda'(u_1)}(\mathsf{i}_{\lambda'(u_2)}(\ldots \mathsf{i}_{\lambda'(u_{|\beta(t)|})}(\mathsf{l}) \ldots)),$$

where $(u_1, u_2, \ldots, u_{|\beta(t)|})$ is an arbitrary ordering of the vertices of $\beta(t)$.

Consider now the case when $p > 0$. For $i = 1, 2, \ldots, p$, let $\mathbf{t}_i$ be equal to $\mathbf{t}_{t_i,\lambda_i}$, where $\lambda_i = \lambda'|_{\sigma(t_i)}$. Existence of $\mathbf{t}_i$ is asserted by the induction hypothesis for the node t_i. Construct $\mathbf{t}_i'$ from $\mathbf{t}_i$ by applying i operation for all the labels that are used in λ', but not in λ_i. Then we can take

$$\mathbf{t}' := \begin{cases} \mathbf{t}_1' & \text{if } p = 1, \\ \mathsf{j}(\mathbf{t}_1', \mathbf{t}_2', \ldots, \mathbf{t}_p') & \text{if } p > 1. \end{cases}$$

28

It is straightforward to see that in both cases $\mathbf{t}'$ constructs $(G[\gamma(t)] \setminus \binom{\beta(t)}{2}), \lambda')$, as claimed.

Now we need to show how to obtain $\mathbf{t}_{t,\lambda}$ from $\mathbf{t}'$. Let L be the set of labels used in λ' and let $L_\sigma \subseteq L$ be the set of labels used in λ. To obtain $\mathbf{t}_{t,\lambda}$ from $\mathbf{t}'$, we perform the following operation: We iterate over labels $i \in L \setminus L_\sigma$ in an arbitrary order, and we apply the $\mathfrak{f}$ operation to i and the set M being equal to the set of labels of the neighbors of the vertex labeled i in the bag $\beta(t)$. It is straightforward to see that $\mathbf{t}_{t,\lambda}$ constructs $(G[\gamma(t)] \setminus \binom{\sigma(t)}{2}), \lambda)$, as claimed. This concludes the inductive proof. $\qquad\square$