HAL open science

Improved Bounds for Twin-Width Parameter Variants with Algorithmic Applications to Counting Graph Colorings

Ambroise Baril, Miguel Couceiro, Victor Lagerkvist

To cite this version:

Ambroise Baril, Miguel Couceiro, Victor Lagerkvist. Improved Bounds for Twin-Width Parameter Variants with Algorithmic Applications to Counting Graph Colorings. 2023. hal-04142719v3

HAL Id: hal-04142719

https://hal.science/hal-04142719v3

Preprint submitted on 13 May 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Improved Bounds for Twin-Width Parameter Variants with Algorithmic Applications to Counting Graph Colorings

Ambroise Baril[1*], Miguel Couceiro[1,3] and Victor Lagerkvist[2]

[1*]LORIA, Université de Lorraine, France.
[2]Department of Computer and Information Science, Linköpings universitet, Sweden.
[3]INESC-ID, IST, Universidade de Lisboa, Portugal.

*Corresponding author(s). E-mail(s): ambroise.baril@loria.fr;
Contributing authors: miguel.couceiro@loria.fr; victor.lagerkvist@liu.se;

Abstract

The $\boldsymbol{H}$-Coloring problem is a well-known generalization of the classical NP-complete problem $\boldsymbol{k}$-Coloring where the task is to determine whether an input graph admits a homomorphism to the template graph $\boldsymbol{H}$. This problem has been the subject of intense theoretical research and in this article we study the complexity of $\boldsymbol{H}$-Coloring with respect to the parameters *clique-width* and the more recent *component twin-width*, which describe desirable computational properties of graphs. We give two surprising linear bounds between these parameters, thus improving the previously known exponential and double exponential bounds. Our constructive proof naturally extends to related parameters and as a showcase we prove that *total twin-width* and *linear clique-width* can be related via a tight quadratic bound. These bounds naturally lead to algorithmic applications. The linear bounds between component twin-width and clique-width entail natural approximations of component twin-width, by making use of the results known for clique-width. As for computational aspects of graph coloring, we target the richer problem of counting the number of homomorphisms to $\boldsymbol{H}$ ($\#\boldsymbol{H}$-Coloring). The first algorithm that we propose uses a contraction sequence of the input graph $\boldsymbol{G}$ parameterized by the component twin-width of $\boldsymbol{G}$. This leads to a positive FPT result for the counting version. The second uses a contraction sequence of the template graph $\boldsymbol{H}$ and here we instead measure the complexity with respect to the number of vertices in the input graph. Using our linear bounds we show that our algorithms are *always* at least as fast as the previously best $\#\boldsymbol{H}$-Coloring algorithms (based on clique-width) and for several interesting classes of graphs

(e.g., cographs, cycles of length $\geq \mathbf{7}$, or distance-hereditary graphs) are in fact strictly faster.

1 Introduction

Graph coloring is a well-known computational problem where the goal is to color a graph in a consistent way. This problem is one of the most well-studied NP-hard problems and enjoys a wide range of applications *e.g.*, in planning, scheduling, and resource allocation [1]. There are many variants and different formulations of the coloring problem, but the most common formulation is certainly the k-Coloring problem that asks whether the vertices of an input graph can be colored using k available colors in such a way that no two adjacent vertices are assigned the same color. This problem can be extended in many ways and in this paper we are particularly interested in the more general problem where any two adjacent vertices in the input graph G have to be mapped to two adjacent vertices in a fixed template graph H (the H-Coloring problem). It is not difficult to see that k-Coloring is then K_k-Coloring, where K_k is the k-vertex clique.

The basic H-Coloring problem has been extended in many directions, of which one of the most dominant formalisms is the *counting* extension where the task is not only to decide whether there is at least one solution (coloring) but to return the number of solutions ($\#H$-Coloring). This framework makes it possible to encode phase transition systems modeled by partition functions, modeling problems from statistical physics such as counting q-particle Widom–Rowlinson configurations and counting Beach models, or the classical Ising model (for further examples, see *e.g.* Dyer & Greenhill [2]). The $\#H$-Coloring problem is #P-hard unless every connected component of H is either a single vertex without a loop, a looped clique or a bipartite complete graph, and it is in P otherwise [2]. The question is then to which degree we can still hope to solve it efficiently, or at least improve upon the naive bound of $|V_H|^{|V_G|}$ (where V_H is the set of vertices in the template graph H and V_G the set of vertices in the input graph G).

In this article we tackle this question by targeting properties of graphs, so-called *graph parameters*, which give rise to efficiently solvable subproblems. We will see below several concrete examples of graph parameters, but for the moment we simply assume that each graph G is associated with a number $k \in \mathbb{N}$, a *parameter*, which describes a structural property of G. Here, the idea is that small values of k correspond to graphs with a simple structure, while large values correspond to more complicated graphs.

There are then two ways to approach intractable H-Coloring problems: we either restrict the class of *input* graphs G, or the class of *template* graphs H to graphs where the parameter is bounded by some reasonably small constant. The first task is typically studied using tools from *parameterized* complexity where the goal is to prove that problems are *fixed-parameter tractable* (FPT), *i.e.*, obtaining running times of the form $f(k) \cdot \|G\|^{O(1)}$ for a computable function $f\colon \mathbb{N} \to \mathbb{N}$ (where $\|G\|$ is the number of bits required to represent the input graph G). The second task is more closely related

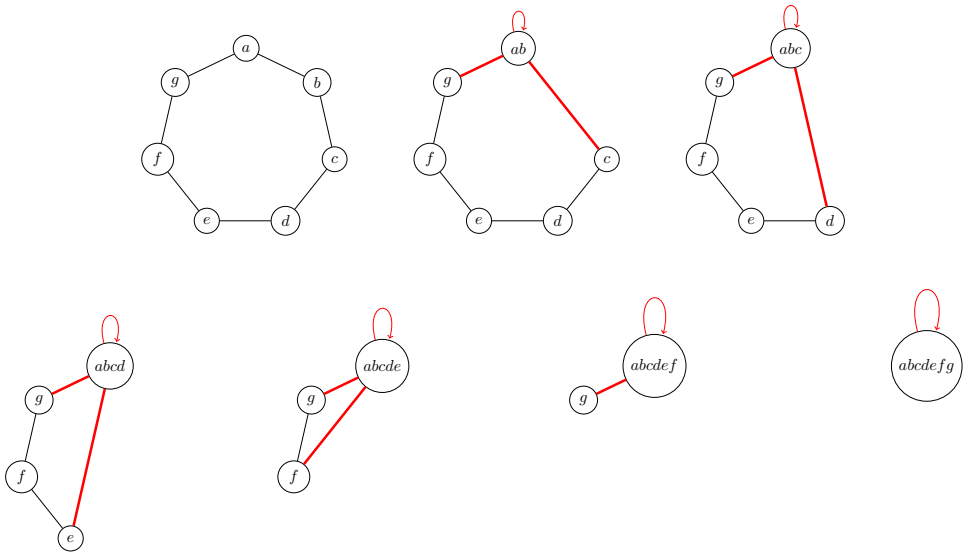

Fig. 1 A contraction sequence of the 7-cycle.

to *fine-grained* complexity[1] where the goal is to prove upper and lower bounds of the form $2^{f(k)} \cdot \|G\|^{O(1)}$ for a sufficiently "fine-grained" parameter k, which in our case is always going to denote the number of vertices $|V_G|$ in the input graph G. Here, it is worth remarking that H-Coloring is believed to be a very hard problem, and the general Coloring problem, where the template is part of the input, is not solvable in $2^{O(|V_G|)} \cdot (\|G\| + \|H\|)^{O(1)}$ time under the *exponential-time hypothesis* (ETH) [4]. Hence, regardless of whether one studies the problem under the lens of parameterized or fine-grained complexity, one needs to limit the class of considered graphs via a suitable parameter.

The most notable graph parameter in this context is likely *treewidth* which, intuitively, measures how close a graph is to being a tree. Bounded treewidth is in many algorithmic applications sufficient to guarantee the existence of an FPT algorithm, but with the shortcoming of failing to capture classes of dense graphs. There are many graph parameters proposed to address this limitation of tree-width, and we briefly survey two noteworthy examples (see Section 2 for formal definitions).

1. *clique-width* (**cw**). The class of graphs (with labelled vertices) with clique-width $\leq k$ is defined as the smallest class of graphs that contains the one vertex graphs $\bullet_i$ with 1 vertex labelled by $i \in [k]$, and that is stable by the following operations for $(i, j) \in [k]^2$ with $i \neq j$: (i) disjoint union of graphs, (ii) relabelling every vertex of label i to label j, and (iii) constructing edges between every vertex labelled by i and every vertex labelled by j. Note that the class of cographs (which contains cliques) is exactly that of graphs with clique-width at most 2.

[1] The upper bound aspect of this field also goes under the name of "exact exponential-time algorithms" [3]. Let us also remark that fine-grained complexity is also strongly associated with proving sharp lower bounds for problems in P.

2. *twin-width* (**tww**). The class of graphs of twin-width $\leq k$ is usually formulated via *contraction sequences* where graphs are gradually merged into a single vertex (see Figure 1 for an example). Red edges represent an inconsistency in the merged vertex (see Section 2.3 for a formal definition), and the maximum red degree in the sequence thus represents the largest loss of information. A graph has twin-width $\leq k$ if it admits such a contraction sequence where the maximum red degree does not exceed k.

For clique-width, Ganian et. al [5] identified a structural parameter s of graphs (the number of distinct non-empty intersections of neighborhoods over sets of vertices), and presented an algorithm for H-Coloring that runs in $O^*(s(H)^{\mathbf{cw}(G)})$ time[2]. It is also optimal in the sense that if there is an algorithm that solves H-Coloring in time $O^*((s(H)-\varepsilon)^{\mathbf{cw}(G)})$, then the SETH fails [5]. Alternative algorithms exist for templates of bounded clique-width, see Wahlström [6] who solves $\#H$-Coloring in $O^*((2\mathbf{cw}(H)+1)^{|V_G|})$ time, and Bulatov & Dadsetan [7] for extensions.

Twin-width, on the other hand, is a much more recent parameter, but has in only a few years attracted significant attention [8–29]. One of its greatest achievements is that checking if a graph is a model of any first-order formula can be decided in FPT time parameterized by the twin-width of the input graph. Thus, a very natural research question in light of the above results concerning tree- and clique-width is to study the complexity of $(\#)H$-Coloring via twin-width. Unfortunately, it is easy to see that under standard assumptions, H-Coloring is generally not FPT parameterized by twin-width. Indeed, since twin-width is bounded on planar graphs [30], the existence of an FPT algorithm for 3-Coloring running in $O^*(f(\mathbf{tww}(G)))$ time implies an $O^*(1)$ time (*i.e.* a polynomial time) algorithm for 3-Coloring on planar graphs (since $f(\mathbf{tww}(G)) = O(1)$ if G is a planar graph). Since 3-Coloring is NP-hard on planar graphs, this would imply P=NP. Thus, 3-Coloring is *para*-NP-*hard* [31] parameterized by twin-width.

Despite this hardness result it is possible to analyze H-Coloring by a variant of twin-width known as *component twin-width* (**ctww**) [15]. This parameter equals the maximal size of a red-connected component (instead of the maximal red-degree for twin-width). It is then known that component twin-width is functionally equivalent[3] to boolean-width [15], which in turn is functionally equivalent to clique-width [32]. Hence, H-Coloring is FPT parameterized by component twin-width, and the specific problem k-Coloring is additionally known to be solvable in $O^*((2^k-1)^{\mathbf{ctww}(G)})$ time [15]. As remarked by Bonnet et al., the theoretical implications of this particular algorithm are limited due to the aforementioned (under the SETH) optimal algorithm parameterized by clique-width [5]. However, this still leaves several gaps in our understanding of component twin-width for H-Coloring and its counting extension $\#H$-Coloring.

Our paper has three major contributions to bridge these gaps. *Firstly,* the best known bounds between clique-width and component twin-width are obtained by following the proof of functional equivalence between component twin-width and

[2] The notation O^* means that we ignore polynomial factors.
[3] *I.e.*, each parameter is bounded by a function of the other.

boolean-width, and then between boolean-width and clique-width. We thereby obtain

$$\mathbf{ctww} \leq 2^{\mathbf{cw}+1} \quad \text{and} \quad \mathbf{cw} \leq 2^{2^{\mathbf{ctww}}}$$

and H-Coloring is thus solvable in $O^*(s(H)^{2^{2^{\mathbf{ctww}(G)}}})$ time. This proves FPT but with a rather prohibitive run-time, and the main question is whether it is possible to improve this to a single-exponential running time $O^*(2^{O(\mathbf{ctww}(G))})$. (This line of research in parameterized complexity is relatively new but of growing importance and has seen several landmark results, see *e.g.* Chapter 11 in Cygan et al. [33]). We prove that it is indeed possible by significantly strengthening the bounds between **cw** and **ctww** and obtain the linear bounds

$$\mathbf{cw} \leq \mathbf{ctww} + 1 \leq 2\mathbf{cw}.$$

Our proof is constructive which gives a fast algorithm to derive a contraction-sequence from a clique-width expression and vice versa. To demonstrate that these ideas are not limited to these specific parameters we (in Section 3.3) consider the related problem of proving tighter bounds between *linear clique-width* (**lcw**) and the recently introduced *total twin-width* (**ttww** [15]). Linear clique-width is less explored than clique-width but comes with the advantage that faster algorithms for graph classes of bounded linear clique-width are sometimes possible (cf. the remark before Theorem 7 in Bodlaender et al. [34]) and that lower bounds on clique-width in many interesting cases can be generalized to linear clique-width [35]. The total twin-width parameter is then known to be functionally equivalent to linear clique-width, yielding the doubly exponential bounds $\mathbf{lcw} \leq 2^{2^{\mathbf{ttww}}+1}$ and $\mathbf{ttww} \leq (2^{\mathbf{lcw}}+1)(2^{\mathbf{lcw}-1}+1)$. We significantly improve the latter to

$$\mathbf{lcw} - 1 \leq 2\mathbf{ttww} \leq \mathbf{lcw}(\mathbf{lcw}+1),$$

and thus demonstrate that virtually any complexity question regarding linear clique-width can be translated to the total twin-width setting, with the possible advantage of using contraction sequences as a unifying lens. Specifically, it can be expected that contraction sequence related parameters are more convenient to use than (linear) clique-width, since there is only one fundamental operation to handle (vertex contraction) whereas (linear) clique-width not only deals with vertex-labelled graphs, but also introduces four fundamental operations.

Secondly, we discuss how these bounds can be exploited to *approximate* **ctww** by making use of the results known on **cw**. Thus, an immediate consequence of our linear bounds is that H-Coloring is solvable in $O^*(s(H)^{\mathbf{ctww}(G)+1})$ time, which is a major improvement to the aforementioned triple exponential upper bound.

Thirdly, we consider the generalized problem of counting the number of solutions. It seems unlikely that the optimal algorithm (under SETH) by Ganian et al. [5] can be lifted to #H-Coloring, and while the algorithm by Wahlström [6] successfully solves #H-Coloring, it does so with the significantly worse bound of $2^{2\mathbf{cw}(G)\times|V_H|}(|V_G|+|V_H|)^{O(1)}$. We tackle this problem in Section 4 by designing a novel algorithm for #H-Coloring for input graphs with bounded component twin-width and which runs in $(2^{|V_H|}-1)^{\mathbf{ctww}(G)} \times (|V_G|+|V_H|)^{O(1)}$ time. Since our linear bounds imply that

$\mathbf{ctww}(H)+2 \leq 2\mathbf{cw}(H)+1$ and $\mathbf{ctww}(H)+2 \leq \mathbf{lcw}(H)+2$ this is always at least as fast as the (linear) clique-width algorithm by Wahlström, and strictly faster for several interesting classes of graphs. For example, cographs with edges (component twin-width 1, versus clique-width 2), cycles of length at least 7 (component twin-width 3, versus linear clique-width 4), and distance hereditary graphs (component twin-width 3 versus clique-width 3 [36]).

We also consider $\#H$-Coloring when the template graph H has bounded component twin-width. We use an optimal contraction sequence of H in order to obtain a $O^*((\mathbf{ctww}(H)+2)^{|V_G|})$ algorithm for $\#H$-Coloring. For comparison, Wahlström [6] solves $\#H$-Coloring in $O^*((2\mathbf{cw}(H)+1)^{|V_G|})$ and, slightly faster, $O^*((\mathbf{lcw}(H)+2)^{|V_G|})$. Due to our linear bounds we again conclude that our algorithm is always at least as fast as the $O^*((2\mathbf{cw}(H)+1)^{|V_G|})$ time clique-width algorithm by Wahlström [6], and strictly faster for the aforementioned classes of graphs. For example, if H is a cograph with edges then our algorithm solves $\#H$-coloring in $O^*(3^{|V_G|})$ time which beats the clique-width $O^*(5^{|V_G|})$ algorithm by a significant margin. Let us also remark that the class of cographs does not have bounded linear clique-width, so the $O^*((\mathbf{lcw}(H)+2)^{|V_G|})$ algorithm is not relevant. Also, if H is a distance-hereditary graph, our algorithm solves $\#H$-coloring in $O^*(5^{|V_G|})$ time which beats the clique-width $O^*(7^{|V_G|})$ algorithm. If H is a cycle of length at least 7 we instead get $\mathbf{ctww}(H)=3$, $\mathbf{cw}(H)=4$, $\mathbf{lcw}(H)=4$, yielding the bounds $O^*(5^{|V_G|})$, $O^*(9^{|V_G|})$, and $O^*(6^{|V_G|})$, *i.e.*, also in this case our algorithm is strictly faster.

Moreover, let us also remark that the technique employed in this article could similarly be used to derive the same results applied to the more general frameworks of counting the solutions of *binary constraint satisfaction problems*, *i.e.*, problems of the forms #Binary-Csp(Γ) with a set of binary relations Γ over a finite domain. However, to simplify the presentation we restrict our attention to the $\#H$-Coloring problem.

2 Preliminaries

Throughout this paper, a *graph* G is a tuple (V_G, E_G), where V_G is a finite set (the set of vertices of G), and E_G is a binary irreflexive symmetric relation over V_G (the set of edges of G). A *looped-graph* is a G is a tuple (V_G, E_G), where V_G is a finite set (the set of vertices of G), and E_G is a binary symmetric relation (not necessarily irreflexive) over V_G (the set of edges of G). We will denote the number of vertices of a graph G by $n(G)$ or, simply, by n when there is no danger of ambiguity. A *cycle* is a graph isomorphic to the graph $C_n = ([n], \{(i,j) \in [n]^2 \mid |i-j| \in \{1, n-1\}\})$ with $n \geq 3$. The neighborhood of a vertex u of a graph G is the set $N_G(u) = \{v \in V_G \mid (u,v) \in E_G\}$. For a graph H we let H-Coloring be the computational problem of deciding whether there exists an homomorphism from an input graph G to H, *i.e.*, whether there exists a function $f\colon V_G \to V_H$ such that $(x,y) \in E_G$ implies that $(f(x), f(y)) \in E_H$. We write $\#H$-Coloring for the associated *counting* problem where we instead wish to determine the exact number of such homomorphisms. As remarked in Section 1, the template graph H can be chosen with great flexibility to model many different types of problems.

2.1 Parameterized complexity

We assume that the reader is familiar with parameterized complexity and only introduce the strictly necessary concepts (we refer to Flum & Grohe [37] for further background). A *parameterized counting problem* is a pair (F, dom) where $F : \Sigma^* \mapsto \mathbb{N}$ (for an alphabet Σ, *i.e.*, a finite set of symbols) and dom is a subset of $\Sigma^* \times \mathbb{N}$. A parameterized counting problem (F, dom) is said to be *fixed-parameter tractable* (FPT) if there exists a computable function $f : \mathbb{N} \to \mathbb{N}$ such that for any instance $(x, k) \in dom$ of F, we can compute $F(x)$ in $f(k) \times \|x\|^{O(1)}$ time. An algorithm with this complexity is said to be an FPT *algorithm*. Note that even though f might be superpolynomial, which is expected if the problem is NP-hard, instances where k is reasonably small can still be efficiently solved.

In practice, when studying FPT algorithms for an NP-hard counting problem, it is very natural to optimize the superpolynomial function f that appears in the complexity of the algorithm solving it. Typically, when dealing with graph problems parameterized by the number of vertices n, an algorithm running in $c^n \times \|x\|^{O(1)}$ will be considered efficient in practice if $c > 1$ is small. This field of research is sometimes referred to as *fine-grained complexity*.

2.2 Clique-width

For $k \geq 1$, let $[k] = \{1, \ldots, k\}$. A *k-labelled graph* G is a tuple (V_G, E_G, ℓ_G), where (V_G, E_G) is a graph and $\ell_G : V_G \to [k]$. For $i \in [k]$ and a k-labelled graph G, denote by $V_G^i = \ell_G^{-1}(\{i\})$ the set of vertices of G of label i. A *k-expression* φ of a k-labelled graph G, denoted $[\varphi] = G$, is an expression defined inductively [38] using:

1. **Single vertex:** $\bullet_i$ with $i \in [k]$: $[\bullet_i]$ is a k-labelled graph with one vertex labelled by i (we sometimes write $\bullet_i(u)$ to state that the vertex is named u),
2. **Disjoint Union:** $\varphi_1 \oplus \varphi_2$: $[\varphi_1 \oplus \varphi_2]$ is the disjoint union of the graphs $[\varphi_1]$ and $[\varphi_2]$.
3. **Relabelling:** $\rho_{i \to j}(\varphi)$ with $(i, j) \in [k]^2$ and $i \neq j$: $[\rho_{i \to j}(\varphi)]$ is the same graph as $[\varphi]$, in which all vertices of G with former label i now have label j,
4. **Edge Creation:** $\eta_{i,j}(\varphi)$ with $(i, j) \in [k]^2$ and $i \neq j$: $[\eta_{i,j}(\varphi)]$ is the same graph as $[\varphi]$, in which all tuples of the form (u, v) with $\{\ell_G(u), \ell_G(v)\} = \{i, j\}$ is now an edge.

A graph G has a k-expression φ if there exists $\ell : V_G \mapsto [k]$ such that $[\varphi] = (V_G, E_G, \ell)$. The *clique-width* of a graph G (denoted by $\mathbf{cw}(G)$) is the minimum $k \geq 1$ such that G has a k-expression. An *optimal expression* of a graph G is a $\mathbf{cw}(G)$-expression of G. The *subexpressions* of an expression φ are defined similarly: the only subexpression of $\bullet_i$ is $\bullet_i$, the subexpressions of $\varphi = \varphi_1 \oplus \varphi_2$ are φ and the subexpressions of φ_1 and φ_2, the subexpressions of $\varphi = \rho_{i \to j}(\varphi')$ and $\varphi = \eta_{i,j}(\varphi')$ are φ and the subexpressions of φ'. A *linear* k-expression is a k-expression φ where for every subexpression of φ of the form $\varphi_1 \oplus \varphi_2$, φ_2 is of the form $\bullet_i$ with $i \in [k]$. The *linear clique-width* (denoted by $\mathbf{lcw}(G)$) of a graph G is the minimum $k \geq 1$ such that G has a linear k-expression.

The most notable of the many graph classes with bounded clique-width is perhaps the class of *cographs*: it is the class of graph that do not contain an induced path

on 4 vertices [39]. The cographs are exactly the graphs of cliquewidth bounded by 2 [40]. Another notable graph class of bounded clique-width is the class of *distance-hereditary graphs*: it is the class of graph in which the distances in any connected induced subgraph are the same as they are in the original graph. The class of distance-hereditary graphs strictly contains the class of cographs, and any distance-hereditary graph has its clique-width bounded by 3 [36].

2.3 Parameters over contraction sequences

Let V be a finite set, and let $n := |V|$. A *partition* of V is a set $\mathcal{P} = \{S_1, \ldots, S_k\}$ (with $k \geq 1$) of non-empty subsets of V, such that every element of V belongs to exactly one of the S_i with $i \in [k]$. A *partition sequence* [15] $(\mathcal{P}_n, \ldots, \mathcal{P}_1)$ of V is a sequence of partitions of V, such that $\mathcal{P}_n$ is the partition into singletons, and each $\mathcal{P}_k$ (with $k \in [n-1]$) is obtained by merging two parts of $\mathcal{P}_{k+1}$: *i.e.* denoting $\mathcal{P}_{k+1} = \{S_1, \ldots, S_{k+1}\}$, there exists $(i, j) \in [k+1]$ with $i \neq j$ and $\mathcal{P}_k = (\mathcal{P}_{k+1} \setminus \{S_i, S_j\}) \cup \{S_i \cup S_j\}$. Note that this definition implies for all $k \in [n]$, that $\mathcal{P}_k$ has k elements, and that in particular, $\mathcal{P}_1 = \{V\}$.

A *trigraph* [41] is a triplet $G = (V_G, E_G, R_G)$ where (V_G, E_G) is a graph and (V_G, R_G) is a looped-graph, with $E_G \cap R_G = \emptyset$. The set E_G is the set of (black) edges of G, and R_G the set of *red edges* of G. The *red-degree* of a vertex $u \in V_G$ is its degree in the looped-graph (V_G, R_G) ignoring the red loops. A *red-connected component* of a trigraph G is a connected component of the looped-graph (V_G, R_G). A trigraph is naturally associated to every partition of the set of vertices of a graph via the following definition.

Definition 1. *Let $G = (V_G, E_G)$ be a graph and $\mathcal{P}$ be a partition of V_G, the trigraph $G/\mathcal{P} = (\mathcal{P}, E_{\mathcal{P}}, R_{\mathcal{P}})$ is defined by :*

- $E_{\mathcal{P}} = \{(S_1, S_2) \in \mathcal{P}^2 \mid S_1 \neq S_2, S_1 \times S_2 \subseteq E_G\}$,
- $R_{\mathcal{P}} = (\{(S_1, S_2) \in \mathcal{P}^2 \mid S_1 \neq S_2, (S_1 \times S_2) \cap E_G \neq \emptyset\} \setminus E_{\mathcal{P}}) \cup \{(S, S) \mid S \in \mathcal{P}, |S| \geq 2\}$.

These choices of definitions for $E_{\mathcal{P}}$ and $R_{\mathcal{P}}$ are strongly motivated by Property 1, that enables to interpret the presence of edges between two different vertices S_1 and S_2 via the bipartite graph induced on G with the bipartition $\{S_1, S_2\}$.

Property 1. *Let G be a graph, $\mathcal{P}$ be a partition of V_G, and let U and V be two different vertices of $G/\mathcal{P}$. For all $u \in U$ and $v \in V$:*

- *$(u, v) \in E_G$, whenever $(U, V) \in E_{G/\mathcal{P}}$, and*
- *$(u, v) \notin E_G$, whenever $(U, V) \notin E_{G/\mathcal{P}} \cup R_{G/\mathcal{P}}$.*

The presence of a black edge indicates a complete bipartite graph, whereas the absence of an edge shows that the bipartite graph has no edge. In contrast, a red edge can be viewed as a loss of information: it will therefore be natural to study parameters that increase with the number of red-edges. The proof of the soundness of our algorithms (in Section 4) that make use of partition sequences rely on this fundamental property. It can be easily obtained by reformulating the definition of partition sequences.

A *contraction sequence* [41] of a graph G on at least two vertices is a sequence of trigraphs of the form $(G_n, \ldots, G_1)$ with $n = |V_G|$, such that there exists a partition

sequence $(\mathcal{P}_n, \ldots, \mathcal{P}_1)$ with for all $k \in [n], G_k = G/\mathcal{P}_k$. If U and V are the elements of $\mathcal{P}_{k+1}$ that are such that $\mathcal{P}_k = (\mathcal{P}_{k+1} \setminus \{U, V\}) \cup \{U \cup V\}$, we write that $G_k = G_{k+1}/(U, V)$, as G_k is obtained from G_{k+1} by contracting the vertices U and V of G_{k+1}. In order to alleviate notations, we will (abusively) denote the vertex $U \cup V$ of G_k as UV. Note that G_k has k vertices and, in particular, the trigraph $G_n = (V_{G_n}, E_{G_n}, \emptyset)$ has no red edge, and the graph (V_{G_n}, E_{G_n}) is isomorphic to G. Note also that G_1 has only one vertex, and is necessarily the trigraph[4] with one vertex $G_1 = (\{V_G\}, \emptyset, \{(V_G, V_G)\})$.

We can remark that a trigraph G_k (with $k \in [n-1]$) obtained in a contraction sequence can be derived easily from G_{k+1}. The rules to follow when performing a contraction are given in Remark 1.

Remark 1. *For each $k \in [n-1]$, the trigraph $G_k = G_{k+1}/(U, V)$ can easily be described in function of the graph G_{k+1}, noticing that for all vertices X and Y of G_k:*

- *If both $X \neq UV$ and $Y \neq UV$, (X, Y) is a black edge (respectively a red edge) in G_k if and only if it is a black edge (respectively red edge) in G_{k+1}.*
- *If $X = Y = UV$, then (X, Y) is a red loop in G_k.*
- *If $X = UV$ and $Y \neq X$, and if both (U, Y) and (V, Y) are black edges in G_{k+1}, then (X, Y) is a black edge in G_k.*
- *If $X = UV$ and $Y \neq X$, and if both (U, Y) and (V, Y) are non-edges (i.e. neither a black edge nor a red edge) in G_{k+1}, then (X, Y) is a non-edge in G_k.*
- *In any other case where $X = UV$ and $Y \neq X$, (X, Y) is a red edge in G_k.*

To define the parameters related to contraction sequences, we introduce various notions of "width" for a trigraph, which is a function assigning an integer to any trigraph. We extend the notion of width to contraction sequences by considering the maximum width of the trigraphs occurring in the sequence. Finally, the width of a graph is defined as the minimum width among all its contraction sequences. Also, if the width notion is clear from the context, we say that a contraction sequence of a graph G is *optimal* if its width equals the width of G.

The *twin-width* (**tww**) [41] of a trigraph is the maximal red-degree of its vertices. Similarly, the *component twin-width* (**ctww**) of a trigraph is the maximal size of a red-connected component. Also, the *total twin-width* (**ttww**)[15] of trigraph is its number of red-edges.

It is known that the class of graph that admits a contraction sequence without red edges (except red loops) is exactly the class of cographs [41]. As a consequence, the cographs are exactly the graphs of twin-width 0, and of component twin-width 1.

We also introduce a new parameter that we call the *total vertex twin-width*. The *total vertex twin-width* (**tvtww**) of a trigraph is its number of vertices adjacent to at least one red edge (including red loops). We believe that this "vertex-based parameter" opens more interesting computational applications than the "edge-based parameter" total twin-width, as it is arguably more natural for algorithms to iterate over vertices than over edges. However, the two parameter are closely connected by natural linear and quadratic bounds. Clearly, if a looped-graph has $t \geq 0$ vertices of degree at least

[4] Each vertex of G_k is a set of vertices of G that have been contracted.

1, it has at least $t/2$ edges and at most $t(t+1)/2$ edges. Applying these remarks to the red graphs $(V_{G'}, R_{G'})$ of a trigraph G' leads to the following quadratic bounds.

Theorem 2. *For any graph G,*

$$\mathbf{tvtww}(G) \leq 2\mathbf{ttww}(G) \leq (\mathbf{tvtww}(G))(\mathbf{tvtww}(G)+1).$$

2.4 Rank-width

A *branch decomposition* [42] of a graph G is a binary tree T (a tree where each non-leaf vertex has degree 3) whose set of leaves is exactly V_G. Let G be a graph and T a branch decomposition of G. Every edge e of T corresponds to a bipartition (X_e, Y_e) of V_G by considering the bipartition of the leaves of T into their connected components of $T - e$ (the tree T but in which the edge e have been removed). For every edge e of T, let A_e be the $\mathbb{F}_2$-matrix whose set of rows is X_e and whose set of columns is Y_e, and whose coefficient of index $(u, v) \in X_e \times Y_e$ is 1 if $(u, v) \in E_G$, and 0 otherwise.

Finally, let $\rho_G(T) = \max_{e \in E_T} \mathbf{rank}(A_e)$. The *rank-width* of G denoted by $\mathbf{rw}(G)$, is the minimum of $\rho_G(T)$ for every branch-decomposition T of G. A branch decomposition T realizing this minimum is called an *optimal* branch-decomposition of G. One of the main interests of rank-width is made clear in the following remark.

Lemma 1. *Let T be an optimal branch-decomposition of a graph G, and $e \in E_T$. If $|X_e| > 2^{\mathbf{rw}(G)}$, then there exists $(u, u') \in (X_e)^2$ with $u \neq u'$ such that*

$$N_G(u) \cap Y_e = N_G(u') \cap Y_e.$$

Proof. Since the rank of the matrix A_e is lower than $\mathbf{rw}(G)$, the rows of G all belong to a $\mathbb{F}_2$-vector space of dimension at most $\mathbf{rw}(G)$. The latter has a cardinality of at most $2^{\mathbf{rw}(G)}$, and therefore, X_e has 2 identical rows, which proves the result. □

3 Improved Bounds for Contraction Sequences Related Parameters

Let us now begin the first major technical contribution of the article. In Section 3.1 we relate component twin-width and clique-width via a tight linear bound. As a consequence, we also manage to relate linear clique-width to component twin-width and show that the component twin-width of a graph is never higher than its linear clique-width. Then, in Section 3.2 we turn to the problem of *approximating* component twin-width (for a given input graph). We show two positive results, one using clique-width as an intermediate parameter, and an improved approximation via rank-width. Lastly, we (in Section 3.3) prove a novel quadratic bound between total twin-width and linear clique-width. Hence, not only can (linear) clique-width be expressed via the twin-width parameter family, but this can be accomplished with a relatively small overhead.

3.1 Comparing clique-width and component twin-width

In this section, we prove the linear bounds between clique-width **cw** and component twin-width **ctww**. As the presence of red-loops does not impact the component twin-width, we ignore them in this section.

Theorem 3. *For every graph G, $\mathbf{cw}(G) \leq \mathbf{ctww}(G) + 1 \leq 2\mathbf{cw}(G)$.*

Firstly, we prove the leftmost inequality. An exemple of the application of the proof of Proposition 4 is provided in Appendix A.1.

Proposition 4. *For every graph G, $\mathbf{cw}(G) \leq \mathbf{ctww}(G) + 1$.*

Proof. Let $(G_n, \dots, G_1)$ be an optimal contraction sequence of G, and let $\kappa = \mathbf{ctww}(G)$. Note that, for all $k \in [n]$, every red-connected component of G_k has size $\leq \kappa$. We explain how to construct a $(\kappa+1)$-expression of G.

We show the following invariant for all $k \in [n]$:
$\mathcal{P}(k)$: *"Let $C = \{S_1, \dots, S_p\}$ be a red-connected component of G_k and $\bigcup C = S_1 \cup \dots \cup S_p$. There exists a $(\kappa+1)$-expression φ_C of the p-labelled graph $G_C = G[\bigcup C]$ with $\forall i \in [p], V^i_{G_C} = S_i$."*

We first prove $\mathcal{P}(n)$. In G_n, there are no red edges: the red-connected components are the singletons $\{u\}$ for $u \in V_G$. Thus $\bullet_1$ is a $(\kappa+1)$-expression of $(G[\{u\}], \ell_u)$ (with $\ell_u : u \mapsto 1$), which proves $\mathcal{P}(n)$.

Now, take $k \in [n-1]$ and assume $\mathcal{P}(k+1)$. We will prove $\mathcal{P}(k)$. By definition of a contraction sequence, G_k is of the form $G_k = G_{k+1}/(U, V)$ for two different vertices U and V of G_{k+1}.

Observe that each red-connected component of G_k is also a red-connected component of G_{k+1}, except the red-connected component C containing UV. Hence, it suffices to prove $\mathcal{P}(k)$ for the red-connected component C. Notice also that $(C \setminus \{UV\}) \cup \{U, V\}$ is a union of red-connected components $C_1, \dots, C_q$ of G_{k+1} (every pair of red-connected vertices in G_{k+1} that does not contain U or V is also red-connected in G_k). We thus have that $C =: (C_1 \cup \dots \cup C_q \cup \{UV\}) \setminus \{U, V\}$.

Denote by $\{S_1, \dots, S_{p-1}, S'_p\}$ the set of vertices of C, with $p = |C|$, and $S'_p = UV$. We have seen that $C_1 \cup \dots \cup C_q = \{S_1, \dots, S_{p-1}, S_p, S_{p+1}\}$, with $S_p := U$ and $S_{p+1} := V$. For each $i \in [p+1]$, S_i belongs to a unique C_j with $j \in [q]$: let $j(i) \in [q]$ be such that $S_i \in C_{j(i)}$.

By $\mathcal{P}(k+1)$ and up to interchanging labels, for every $j \in [q]$ there exists a $(\kappa+1)$-expression φ_{C_j} of the p-labelled graph $G_{C_j} = G[\bigcup C_j]$ with for all $i \in [p]$ with $j(i) = j$, $V^i_{G_{C_j}} = S_i$. Therefore, $\varphi' := \varphi_{C_1} \oplus \dots \oplus \varphi_{C_q}$ expresses the disjoint union of the graphs $G_{C_1}, ..., G_{C_q}$. Furthermore, φ' is an expression of a graph over the same vertices as $G[\bigcup C]$, Now, we still need to construct the black edges crossing these red-connected components.

We thus apply $\eta_{i,i'}$ (edge creation)[5] to φ' for every black edge of the form $(S_i, S_{i'})$ in G_{k+1}, to obtain an expression φ''. Since the vertices with labels i and i' are exactly the vertices of S_i and $S_{i'}$, we create exactly the edges between vertices of S_i and of $S_{i'}$ when applying $\eta_{i,i'}$. By Property 1, we only construct correct black edges in $G[\bigcup C]$, and thus φ'' is an expression of $G[\bigcup C]$. Conversely, as $\mathcal{P}(k+1)$ ensures that $\varphi_{C_1}, \dots, \varphi_{C_q}$ represent exactly $G_{C_1}, \dots, G_{C_q}$, we have that the edges of $G[\bigcup C]$ that

[5] See Section 2.2 for the notations relative to clique-width.

are not represented in φ' are exactly the edges crossing the red-connected components $C_1, \ldots, C_q$ of G_{k+1}. In other words, the edges missing in φ' are necessarily of the form $(a, b) \in S_i \times S_{i'}$, where S_i and $S_{i'}$ do not belong to the same red-connected component. Since $(S_i, S_{i'})$ is not a red edge of G_{k+1} and since $(a, b) \in E_G \cap (S_i \times S_{i'})$, we conclude by Definition 1 that $(S_i, S_{i'})$ is a black edge of G_{k+1}. Thus, $\eta_{i,i'}$ has been applied when constructing φ'', constructing thereby the edge (a, b) in φ''.

Moreover, we need to make sure that the labels in φ'' match the requirements of $\mathcal{P}(k)$. For that, we set $\varphi_{G_C} := \rho_{p+1\to p}(\varphi'')$ (relabelling). By doing so, S_p (say, U) and S_{p+1} (say, V) have the same label in φ_{G_C}. Thus, it follows that φ_{G_C} witnesses $\mathcal{P}(k)$ (since $S_p = U$ and $S_{p+1} = V$ are now contracted into $S'_p = UV$ in G_k) for the red-connected component C. Indeed, we have used $p+1 = |C|+1 \leq \kappa+1$ different labels to construct φ_{G_C} from $\varphi_{C_1}, \ldots, \varphi_{C_q}$. Since $\{V_G\}$ is a red-connected component of G_1, it follows from $\mathcal{P}(1)$ that $G[V_G] = G$ has a $(\kappa+1)$-expression, and thus $\mathbf{cw}(G) \leq \kappa+1$. As $\kappa = \mathbf{ctww}(G)$, we have $\mathbf{cw}(G) \leq \mathbf{ctww}(G) + 1$. □

Note, however, that **lcw** can not be bounded by a function of **ctww**. For instance, the class of cographs have unbounded linear clique-width [43], despite having a bounded component twin-width of 1. Let us now continue by proving the rightmost bound of Theorem 3. An exemple of the application of the proof of Proposition 5 is provided in Appendix A.2.

Proposition 5. *For every graph G, we have:*

(i) $\mathbf{ctww}(G) \leq 2\mathbf{cw}(G) - 1$, *and*
(ii) $\mathbf{ctww}(G) \leq \mathbf{lcw}(G)$.

Proof. We first prove *(i)* and then adapt it to prove *(ii)*. Let $k := \mathbf{cw}(G)$ and take a k-expression of G. We will explain how to construct a contraction sequence of G in which every red-connected component has size $\leq 2k-1$. The following remark will be implicitly used throughout this proof.

Remark 2. *Two vertices that have the same label in an expression φ' also have the same label in any expression of φ that has φ' as a sub-expression.*

We prove the following property of k-expressions of φ by structural induction:
$\mathcal{H}(\varphi)$: *"Let $(G, \ell_G) := [\varphi]$. There exists a (partial) contraction sequence $(G_n, \ldots, G_{k'})$ with $k' \leq k$ of G such that:*

- *every red-connected component in the trigraphs $G_n, \ldots, G_{k'}$ has size $\leq 2k-1$,*
- *the vertices of $G_{k'}$ are exactly the non-empty V_G^i for $i \in [k]$, and*
- *every pair of vertices contracted have the same labels in (G, ℓ_G)[6]."*

If $\varphi = \bullet_i$ with $i \in [k]$, there is nothing to do since G has only one vertex. If φ is of the form $\rho_{i\to j}(\varphi')$ (with $(i, j) \in [k]^2$ and $i \neq j$), consider for G the partial contraction sequence of $(G', \ell_{G'}) := [\varphi']$ given by $\mathcal{H}(\varphi')$, and then contract $V_{G'}^i$ and $V_{G'}^j$ to obtain $V_G^j = V_{G'}^i \cup V_{G'}^j$. Since φ' is also a k-expression of G, and since that last contraction happens in a trigraph with at most k vertices, this partial contraction sequence of G satisfies $\mathcal{H}(\varphi)$.

[6] Inductively, we say that the label of a vertex $S \in V_{G_l}$ $(k' \leq \ell \leq n)$ is then the common label of the vertices that have been contracted together to produce S.

If φ is of the form $\eta_{i,j}(\varphi')$ (with $(i,j) \in [k]^2$ and $i \neq j$), consider for G the partial contraction sequence of $(G', \ell_{G'}) := [\varphi']$ given by $\mathcal{H}(\varphi')$. To prove that it is sufficient to prove $\mathcal{H}(\varphi)$, it is sufficient to justify that it does not create any red edge in the contraction of G that was not present in the contraction of G'. The first red-edge (x, y) that would appear in the contraction of $G = [\eta_{i,j}(\varphi')]$ that does not appear in the same contraction of $G' = [\varphi']$, results necessarily of the contraction of two vertices u and v with $x = uv$ and y being in the symmetric difference of the neighborhoods of u and v in $G = [\eta_{i,j}(\varphi')]$ but not in $G' = [\varphi']$. Such a red-edge can not exist because we contract only vertices with the same label in φ' (or, equivalently, in φ), and that $\eta_{i,j}$ can only decrease (with respect to $\subseteq$) the symmetric difference between the neighborhood of vertices with the same label in φ. By Remark 2, this implies that it is also true for vertices having the same label in any subexpression of φ.

If φ is of the form $\varphi = \varphi' \oplus \varphi''$: denote $(G', \ell') := [\varphi']$ and $(G'', \ell'') := [\varphi'']$, thereby, $V_G = V_{G'} \cup V_{G''}$. Consider the partial contraction sequence of G given by:

1. contract the vertices in $V_{G'}$ in accordance to the contraction sequence given by $\mathcal{H}(\varphi')$,
2. contract the vertices in $V_{G''}$ in accordance to the contraction sequence given by $\mathcal{H}(\varphi'')$,
3. for all $i \in [k]$, contract $V^i_{G'}$ with $V^i_{G''}$ (if both are nonempty) to get $V^i_G = V^i_{G'} \cup V^i_{G''}$.

Steps 1 and 2 do not create a red-edge adjacent to both $V_{G'}$ and $V_{G''}$ (since these are two distinct connected components of G). Thus, before step 3, we have a trigraph with $\leq 2k$ vertices (because both trigraphs obtained after $\mathcal{H}(\varphi')$ and $\mathcal{H}(\varphi'')$ have less than k vertices), and every red-component that have appeared so far has size $\leq 2k-1$. After the first contraction of step 3, the resulting trigraph has $\leq 2k-1$ vertices, and thus no red-connected component of size $> 2k-1$ can emerge. Such a contraction satisfies every requirement of $\mathcal{H}(\varphi)$. We have thus proven $\mathcal{H}(\varphi)$ for every k-expression.

Now, take a k-expression φ of G. Up to applying $\rho_{i\to 1}$ for all $i \in [k]$ to φ, we can assume that $(G, \ell_G) := [\varphi]$ with ℓ_G being constant equal to 1. The partial contraction sequence of G given by $\mathcal{H}(\varphi)$ is a total contraction sequence of G of component twin-width $\leq 2k-1$. Since $k = \mathbf{cw}(G)$, we have proven that $\mathbf{ctww}(G) \leq 2\mathbf{cw}(G) - 1$. To prove (ii), we show a similar property $\mathcal{H}_{\text{lin}}(\varphi)$ for every linear k-expression. The only difference between $\mathcal{H}_{\text{lin}}$ and $\mathcal{H}$ is that we replace the condition $\leq 2k-1$ (on the size of red components) by $\leq k$. The proof then follows exactly the same steps, except for the case $\varphi = \varphi' \oplus \varphi''$, where step 2 (the contraction according to $\mathcal{H}_{\text{lin}}(\varphi'')$) is not necessary anymore, since φ'' is of the form $\bullet_i$ ($i \in [k]$), and we obtain a trigraph of size $k+1$ instead of $2k$, since φ'' has 1 vertex instead of k. This ensures that every red-connected component has size $\leq (k+1) - 1 = k$ instead of $2k-1$ in the non-linear case.

For step 3, *i.e.,* contracting vertices of the same color in φ' and in φ'', just note that it consists of at most 1 contraction instead of k in the linear case. □

We see that the linearity of a k-expression enables to derive a stronger upperbound on the component twin-width of the graph it represents. Note that more generally, if for all subexpression of φ of the form $\varphi_1 \oplus \varphi_2$, the sum of the number of labels in φ_1 and in φ_2 does not exceed an integer $t \geq 2$, we can conclude with the same routine that

$\mathbf{ctww}(G) \leq t-1$. This observation leads to a tight upper bound on the component twin-width of distance-hereditary graphs.

Remark 3. *Let G be a distance-hereditary graph. We have* $\mathbf{ctww}(G) \leq 3$.

Indeed, if G is a distance-hereditary graph, Golumbic and Rotics [36] witness that $\mathbf{cw}(G) \leq 3$ by providing a 3-expression φ of that is such that, for every subexpression of φ of the form $\varphi_1 \oplus \varphi_2$, only 2 different labels occur in φ_1 and in φ_2.

3.2 Approximating component twin-width

The linear bounds established in Section 3.1 entail reasonable approximation results for component twin-width by making use of known approximations of clique-width [44]. The best currently known approximation algorithm for clique-width is given by Theorem 6.

Theorem 6. *[44] Let k be a fixed positive integer. There is an $O(|V_G|^3)$-time (FPT parameterized by k) algorithm that either outputs an (8^k-1)-expression of an input graph G or confirms that the clique-width of G is larger than k.*

From Theorem 6 and the linear bounds established in Proposition 4 and Proposition 5, we immediately obtain an approximation algorithm for component twin-width.

Theorem 7. *Let p be a fixed positive integer. There is an $O(|V_G|^3)$-time (FPT parameterized by k) algorithm that either outputs a contraction sequence of component twin-width $\leq 2^{3p+4}-3$ of an input graph G or confirms that the component twin-width of G is larger than p.*

Proof. The algorithm consists of applying the algorithm of Theorem 6 to G with $k := p+1$. If the algorithm confirms that $\mathbf{cw}(G) > p+1$, then we know that $\mathbf{ctww}(G) > p$ by Proposition 4. Otherwise, it outputs a $(2^{3(p+1)}-1)$-expression of G, which we transform into a contraction sequence of G of component twin-width $\leq 2 \times (2^{3(p+1)}-1)-1 = 2^{3p+4}-3$ through the constructive proof of Proposition 5, which can be performed in linear time in the size of the $(2^{3(p+1)}-1)$-expression of G. □

It is still interesting to see that a direct comparison between component twin-width and rank-width yields to a better approximation ratio, thanks to Theorem 8. In fact, Theorem 6 was also obtained by this method. By avoiding using clique-width as an intermediate parameter, it is not surprising that we obtain a better ratio.

Theorem 8. *[44] Let k be a fixed positive integer. There is an $O(|V(G)|^3)$-time (FPT parameterized by k) algorithm that either outputs a rank-decomposition (of an input graph G) of width at most $3k-1$ or confirms that the rank-width of G is larger than k.*

We can indeed make use of the bounds given by Theorem 9. The proof is very similar to the proof of functional equivalence between boolean-width and component twin-width [15], which is not surprising, since both rely exclusively on Lemma 1, that applies both to rank-width and boolean-width.

Theorem 9. *For every graph G,* $\mathbf{rw}(G)-1 \leq \mathbf{ctww}(G) \leq 2^{\mathbf{rw}(G)+1}$.

Proof. The first inequality $\mathbf{rw}(G)-1 \leq \mathbf{ctww}(G)$ follows from Proposition 4 , stating that $\mathbf{cw}(G)-1 \leq \mathbf{ctww}(G)$ and the fact that $\mathbf{rw}(G) \leq \mathbf{cw}(G)$ [42].

We now focus on proving the second bound of $\mathbf{ctww}(G) \leq 2^{\mathbf{rw}(G)+1}$. This proof follows the same scheme as the proof of the functional equivalence between boolean-width and component twin-width [15].

Similarly to a branch-decomposition of graphs, a branch-decomposition of a tri-graph G' is a binary tree whose set of leaves is $V_{G'}$. It is said to be rooted if a non-leaf vertex has been chosen to be the root, which leads to the usual definition of children and descendants in a rooted tree. The set of leaves descending from a vertex v of a tree T is denoted by D_v. Now, let G be a graph and T be an optimal branch decomposition of G, and let $r := \mathbf{rw}(G)$. We prove by induction the following invariant for $k \in [n]$.

$\mathcal{P}(k)$: *"There exists a (partial) contraction sequence $(G_n, \ldots, G_k)$ of G of component twin-width $\leq 2^{r+1}$. Moreover, there exists a branch-decomposition T_k of G_k such that for every $t \in V_{T_k}$ with $|D_t| > 2^r$, there is no red-edge crossing the bipartition $(D_t, V_{G_k} \setminus D_t)$."*

Note that $\mathcal{P}(n)$ is indeed true since $G = G_n$ has no red-edge. Now assume $\mathcal{P}(k+1)$ with $k \in [n-1]$. We will prove $\mathcal{P}(k)$. First, note that if $k \leq 2^r$, contracting any two arbitrary vertices and giving any branch decomposition of G_k proves $\mathcal{P}(k)$. We may thus assume that $k > 2^r$. The root ρ satisfies $|D_\rho| \geq 2^r + 1$. Observe that there exists a node v of T_{k+1} such that $2^r + 1 \leq |D_v| \leq 2^{r+1}$: a node v such that D_v has size at least $2^r + 1$ and which is furthest from the root meets the condition. Moreover, any child w of v verifies $|D_w| \leq 2^r$. Using Lemma 1 with respect to the edge linking v to its father[7], there are two vertices U and U' of G_{k+1} that satisfy $N_G(U) \cap (V_{G_{k+1}} \setminus D_v) = N_G(U') \cap (V_{G_{k+1}} \setminus D_v)$. Here, the neighborhood are taken with respect to the black edges only, as by $\mathcal{P}(k+1)$, there is no red edge crossing the bipartition $(D_v, V_{G_{k+1}} \setminus D_v)$.

To prove $\mathcal{P}(k)$, we will prove that it is sufficient to contract the vertices U and U' of G_{k+1} to obtain G_k, and to identify the leaves U and U' of T_{k+1} to obtain T_k (*i.e.* we remove u' and shortcut every degree 2 vertex that appears, and we then rename U as UU'). Note that all the red-edges created by the contraction of U and U' are adjacent to the new vertex UU'.

Firstly, by our choice of U and U', we do not create any red-edge crossing $(D_v, V_{G_k} \setminus D_v)$. Due to the property of T_{k+1} ensured by $\mathcal{P}(k+1)$ (recall that $|D_v| > 2^r$), there is no red-edge crossing $(D_v, V_{G_k} \setminus D_v)$ in T_k. The red-connected component C of the new vertex UU' is thus contained in D_v, and thus has size at most 2^{r+1}. Since C is the only red-connected component of G_k that was not a red-connected component of G_{k+1}, G_k indeed meets the requirements of $\mathcal{P}(k)$.

Secondly, due to the choice of v, any node t of T_k with $|D_t| > 2^r$ containing the new vertex UU' is an ancestor of v. Since $D_v \subseteq D_t$, by the above argument, there is no red-edge crossing $(D_t, V_{G_k} \setminus D_t)$.

The proof of $\mathcal{P}(k)$ is now complete: $\mathcal{P}(1)$ justifies that $\mathbf{ctww}(G) \leq 2^{r+1}$. □

This bound naturally leads to the approximation given in Theorem 10.

Theorem 10. *Let p be a fixed positive integer. There is an $O(|V_G|^3)$-time algorithm that either outputs a contraction sequence of component twin-width $\leq 8^{p+1}$ of an input graph G or confirms that the component twin-width of G is larger than p.*

[7] If v is the root, the result is trivial.

Proof. This result can be obtained similarly to Theorem 7, by using Theorem 8 and Theorem 9 instead of Theorem 3 and Theorem 6. □

3.3 Comparing total twin-width and linear clique-width

In this section, we provide a quadratic bound between total twin-width and linear clique-width. As discussed in Section 1 these parameters are known to be functionally equivalent, since they are both known to be functionally equivalent to *linear boolean-width* through the following relations [15, 45]:

- $\mathbf{lbw} \leq \mathbf{lcw} \leq 2^{\mathbf{lbw}+1}$,
- $\mathbf{lbw} \leq 2^{\mathbf{ttww}}$,
- $\mathbf{ttww} \leq (2^{\mathbf{lbw}} + 1)(2^{\mathbf{lbw}-1} + 1)$,

which entail the exponential and double-exponential bounds between linear clique-width and total twin-width:

- $\mathbf{ttww} \leq (2^{\mathbf{lcw}} + 1)(2^{\mathbf{lcw}-1} + 1)$,
- $\mathbf{lcw} \leq 2^{2^{\mathbf{ttww}}+1}$.

These exponential and double exponential bounds are similar to the bounds known between component twin-width and clique-width presented in Section 1.

Here, we will improve these bounds by establishing:

Theorem 11. *For every graph G,* $\mathbf{lcw}(G) - 1 \leq 2\mathbf{ttww}(G) \leq \mathbf{lcw}(G)(\mathbf{lcw}(G) + 1)$.

The proof technique mirrors those of Proposition 4 and Proposition 5. Hence, our proof constructions appear to be generally applicable for showing stronger relationships between graph parameters than mere functional equivalence. We begin by first comparing linear clique-width and total vertex twin-width, and then use Theorem 2. As we will prove, the parameter **tvtww** is exactly the same as **lcw** (up to a difference of 1).

Theorem 12. *For every graph G,* $\mathbf{lcw}(G) - 1 \leq \mathbf{tvtww}(G) \leq \mathbf{lcw}(G)$.

Firstly, we show the leftmost inequality. An exemple of the application of the proof of Proposition 13 is provided in Appendix A.1.

Proposition 13. *For every graph G,* $\mathbf{lcw}(G) \leq \mathbf{tvtww}(G) + 1$.

Proof. The proof is similar to the proof of Proposition 4 but we include the details since the proof is constructive and has potential algorithmic applications. Let $(G_n, \ldots, G_1)$ be a contraction sequence of G witnessing $\kappa := \mathbf{tvtww}(G)$. We explain how to construct a linear $(\kappa + 1)$-expression of G. We show the following invariant for all $k \in [n]$:

$\mathcal{P}(k)$: *"Let $C_k = \{S_1, \ldots, S_p\}$ be the set of vertices of G_k of red-degree at least 1, and $\bigcup C_k = S_1 \cup \cdots \cup S_p$. There exists a linear $(\kappa + 1)$-expression φ_{C_k} of the p-labelled graph $G_{C_k} := G[\bigcup C_k]$ with $V^i_{G_{C_k}} = S_i$ for all $i \in [p]$."*

Note that for all $k \in [n]$, $|C_k| \leq \kappa$ by definition of the total vertex twin-width. We first prove $\mathcal{P}(n)$. In G_n, there are no red edges. Thus, $C_n = \emptyset$ and there is nothing to prove.

Now, take $k \in [n-1]$ and assume $\mathcal{P}(k+1)$. We will prove $\mathcal{P}(k)$. By definition of a contraction sequence, G_k is of the form $G_k = G_{k+1}/(U, V)$ for two different vertices

U and V of G_{k+1}. First, we need to build a linear $(\kappa+1)$-expression over the right set of vertices. Denote $C_k = \{S_1, \ldots, S_{p-1}, S'_p\}$ with $S'_p = UV$. Letting $S_p = U$ and $S_{p+1} = V$, we have that S_i is a vertex of G_{k+1} for all $i \in [p+1]$, and that

$$\bigcup C_k = \bigcup_{i=1}^{p+1} S_i.$$

Observe that, C_{k+1} is of the form $\{S_i \mid i \in I\}$ with $I \subseteq [p+1]$. Also, the other vertices S_j with $j \in [p+1] \setminus I$ of G_{k+1} are necessarily singletons. Otherwise, these vertices would have a red loop in G_{k+1} (by Definition 1) and would thus belong to C_{k+1}. For all $j \in [p+1] \setminus I$, let $S_j = \{s_j\}$ with $s_j \in V_G$.

By $\mathcal{P}(k+1)$, up to interchanging labels, there exists a linear $(\kappa+1)$-expression $\varphi_{C_{k+1}}$ of the $|I|$-labelled graph $G_{C_{k+1}}$, such that for all $i \in I$, $V^i_{G_{C_{k+1}}} = S_i$. Therefore,

$$\varphi' := \varphi_{C_{k+1}} \oplus \bigoplus_{j \in [p+1] \setminus I} \bullet_j (s_j)$$

is a linear expression over the same vertices of the graph G_{C_k}, that satisfies $V^i_{[\varphi']} = S_i$ for all $i \in [p+1]$.

Now, we still need to construct the black edges crossing the different S_i for $i \in [p+1]$. We thus apply $\eta_{i,i'}$[8] to φ' for every black edge of the form $(S_i, S_{i'})$ in G_{k+1} (with $(i, i') \in [p+1]$), to obtain an expression φ''. Since the vertices with labels i and i' are exactly the vertices of S_i and $S_{i'}$, we create exactly the edges between vertices of S_i and of $S_{i'}$ when applying $\eta_{i,i'}$ (the reasioning is similar as in the proof of Proposition 4). By Property 1, and because $\varphi_{C_{k+1}}$ is a linear expression of $G_{C_{k+1}}$, we have that φ'' is a linear expression of G_{C_k}.

Moreover, we need to make sure that the labels in φ'' match the requirements of $\mathcal{P}(k)$. For that, we set $\varphi_{G_{C_k}} := \rho_{p+1 \to p}(\varphi'')$. By doing so, S_p (say, U) and S_{p+1} (say, V) have the same label in $\varphi_{G_{C_k}}$.

Thus, it follows that $\varphi_{G_{C_k}}$ witnesses $\mathcal{P}(k)$ (since $S_p = U$ and $S_{p+1} = V$ are now contracted into $S'_p = UV$ in G_k). Indeed, we have used $p+1 = |C_k|+1 \leq \kappa+1$ different labels to construct the linear expression $\varphi_{G_{C_k}}$. The expression $\varphi_{G_{C_k}}$ is indeed linear because $\varphi_{G_{C_{k+1}}}$ is linear and because the right term of every $\oplus$ used to construct φ_{C_k} from $\varphi_{C_{k+1}}$ is of the form $\bullet_j(s_j)$ with $s_j \in V_G$.

Since $\{V_G\}$ is a vertex of G_1 with a red loop (unless G is a graph on 1 vertex, in which case the theorem is trivial), it follows from $\mathcal{P}(1)$ that $G[V_G] = G$ has a linear $(\kappa+1)$-expression, and thus $\mathbf{lcw}(G) \leq \kappa+1$. As $\kappa = \mathbf{tvtww}(G)$, we have $\mathbf{lcw}(G) \leq \mathbf{tvtww}(G) + 1$. □

We now prove the rightmost bound of Theorem 12.

Proposition 14. *For every graph G, we have,* $\mathbf{tvtww}(G) \leq \mathbf{lcw}(G)$

Proof. Again, we remark that the proof is similar to the proof of Proposition 5, but we include the details since the proof of the contraction sequence with the necessary properties is constructive and may be useful in its own right.

[8] See Section 2.2 for the notations relative to clique-width.

Let $k := \mathbf{lcw}(G)$ and take a linear k-expression φ_G of G. We will explain how to construct a contraction sequence of G in which every trigraph has at most k vertices of red degree at least 1. We begin by defining the following property and then prove it by induction over φ:
$\mathcal{H}(\varphi)$: "Let $(G, \ell_G) := [\varphi]$. There exists a (partial) contraction sequence $(G_n, \dots, G_{k'})$ of G with $k' \leq k$ such that:

- each of the trigraphs $G_n, \dots, G_{k'}$ have at most k vertices with red degree ≥ 1,
- the vertices of $G_{k'}$ are exactly the non-empty V_G^i for $i \in [k]$, and
- every pair of vertices contracted have the same labels in (G, ℓ_G)[9]."

If $\varphi = \bullet_i$ with $i \in [k]$, there is nothing to do since G has only one vertex. If φ is of the form $\rho_{i \to j}(\varphi')$ (with $(i, j) \in [k]^2$ and $i \neq j$), consider for G the partial contraction sequence of $(G', \ell_{G'}) := [\varphi']$ given by $\mathcal{H}(\varphi')$, and then contract $V_{G'}^i$ and $V_{G'}^j$ to obtain $V_G^j = V_{G'}^i \cup V_{G'}^j$. Since φ' is also a k-expression of G, and since that last contraction happens in a trigraph with less than k vertices, this partial contraction sequence of G satisfies $\mathcal{H}(\varphi)$.

If φ is of the form $\eta_{i,j}(\varphi')$ (with $(i, j) \in [k]^2$ and $i \neq j$), consider for G the partial contraction sequence of $(G', \ell_{G'}) := [\varphi']$ given by $\mathcal{H}(\varphi')$. To prove that it is sufficient to prove $\mathcal{H}(\varphi)$, it is sufficient to justify that it does not create any red edge in the contraction of G that was not present in the contraction of G'. The first red-edge (x, y) that would appear in the contraction of $G = [\eta_{i,j}(\varphi')]$ that does not appear in the same contraction of $G' = [\varphi']$, results necessarily of the contraction of two vertices u and v with $x = uv$ and y being in the symmetric difference of the neighborhoods of u and v in $G = [\eta_{i,j}(\varphi')]$ but not in $G' = [\varphi']$. Such a red-edge can not exist because we contract only vertices with the same label in φ' (or, equivalently, in φ), and that $\eta_{i,j}$ can only decrease (with respect to $\subseteq$) the symmetric difference between the neighborhood of vertices with the same label in φ. By Remark 2, this implies that it is also true for vertices having the same label in any subexpression of φ.

If φ is of the form $\varphi = \varphi' \oplus \bullet_i(u)$: denote $(G', \ell') := [\varphi']$, thereby, $V_G = V_{G'} \cup \{u\}$. Consider for G the partial contraction sequence obtained by performing the contractions in $(G', \ell_{G'}) := [\varphi']$ given by $\mathcal{H}(\varphi')$, and then contracting $V_{G'}^i$ (if not empty) and u to obtain $V_G^i = V_{G'}^i \cup \{u\}$. Since u is an isolated vertex in G, performing the contractions in G' can not create any red edge in the contraction of G that did not already exist in the contraction of G'. The last eventual contraction between u and $V_{G'}^i$ occurs in a trigraph with at most $k+1$ vertices, resulting in a trigraph of at most k vertices. In particular, there can not be more than k vertices adjacent to at least one red edge. Such a contraction satisfies every requirement of $\mathcal{H}(\varphi)$. We have thus proven $\mathcal{H}(\varphi)$ for every linear k-expression.

Now, take a linear k-expression φ of G. Up to applying $\rho_{i \to 1}$ for all $i \in [k]$ to φ, we can assume that $(G, \ell_G) := [\varphi]$ with ℓ_G being constant equal to 1. The partial contraction sequence of G given by $\mathcal{H}(\varphi)$ is a total contraction sequence of G of total vertex twin-width $\leq k$. Since $k = \mathbf{lcw}(G)$, we have proven that $\mathbf{tvtww}(G) \leq \mathbf{lcw}(G)$. □

[9] Inductively, we say that the label of a vertex $S \in V_{G_l}$ $(k' \leq l \leq n)$ is then the common label of the vertices that have been contracted together to produce S.

From Theorem 2 and Theorem 12 we then obtain the quadratic bound

$$\mathbf{lcw} - 1 \leq 2\mathbf{ttww} \leq \mathbf{lcw}(\mathbf{lcw} + 1)$$

of Theorem 11.

4 Complexity Results

In the second part of the article we show two algorithmic applications of dynamic programming over component twin-width to $\#H$-Coloring. Let us remark that the proof of Proposition 4 deals with component twin-width with a dynamic programming principle in the following way.

- We keep track of an invariant (here, a clique-width expression) associated to every red connected component.
- The "size of the invariant" (the number of labels) grows with the number of vertices in the component.
- The difficulty of keeping track of the invariant though a contraction is overcame by Property 1, that gives precise information on the structure of the edges intersecting two different red components.

We see in this section how this idea can be used to design dynamic programming algorithm in order to solve counting versions of graph coloring problems. The first result assumes that an optimal contraction sequence of the input graph G is given, and results in a FPT algorithm parameterized by **ctww**, running in time $O^*((2^{|V_H|} - 1)^{\mathbf{ctww}(G)})$. The second approach uses an optimal contraction sequence of the template H (whose computation can be seen as a pre-computation, since it does not involve the input graph G): we obtain a fine-grained algorithm running in time $O^*((\mathbf{ctww}(H) + 2)^{|V_G|})$, which outperforms the best algorithms in the literature, with a running time of $O^*((2\mathbf{cw}(H) + 1)^{|V_G|})$ [6] and $O^*((\mathbf{lcw}(H) + 2)^{|V_G|})$ [6] through the linear bound of Section 3.1. Note that the technique employed in this paper could similarly be used to derive the same complexity results applied to the more general frameworks of counting the solutions of *binary constraint satisfaction problems*, *i.e.* problems of the forms #Binary-Csp(Γ) with Γ a set of binary relations over a finite domain, even though we restrict to the simpler case of $\#H$-Coloring here to avoid having to define contraction sequences of instances and template of binary constraint satisfaction problems.

4.1 Parameterized complexity

We present an algorithm solving $\#H$-Coloring in FPT time parameterized by component twin-width, assuming that a contraction sequence is part of the input. It is inspired by the algorithm solving k-Coloring [15], thus proving that $\#H$-Coloring is FPT parameterized by component twin-width and thus also by clique-width (by functional equivalence). Throughout, we need to assume that we are given a contraction sequence of the input graph.

Let us remark that Walhström [6] solves H-Coloring in time

$$2^{2\mathbf{cw}(G)\times|V_H|}(|V_G|+|V_H|)^{O(1)},$$

whenever a $\mathbf{cw}(G)$-expression of G is given. We solve it in time

$$(2^{|V_H|}-1)^{\mathbf{ctww(G)}+1}\times(|V_G|+|V_H|)^{O(1)}.$$

Recall however that (1) $\mathbf{ctww}(G)+1\leq 2\mathbf{cw}(G)$ by Proposition 5, implying that our algorithm is always at least as fast, and that (2) our algorithm is strictly faster for *e.g.* cographs with edges (component twin-width 1, versus clique-width 2), cycles of length at least 7 (component twin-width 3, versus clique-width 4), and distance-hereditary graphs that are not cographs (component twin-width ≤ 3 by Remark 3, clique-width 3).

Theorem 15. *For any graph H, there exists an algorithm running in time*

$$(2^{|V_H|}-1)^{\mathbf{ctww(G)}+1}\times(|V_G|+|V_H|)^{O(1)}$$

that solves $\#H$-Coloring on any input graph G (assuming that an optimal contraction sequence $(G_n,\ldots,G_1)$ of G is given).

Proof. For $k\in[n]$, $C=\{S_1,\ldots,S_p\}\subseteq V_{G_k}$ a red-connected component of vertices of G_k, and for $\gamma:C\mapsto(2^{V_H}\setminus\{\emptyset\})$, an *$H$-coloring of $G[\cup C]$ with profile γ* is an H-coloring f of $G[\cup C]$ such that for all $i\in[p]$, $f(S_i)=\gamma(S_i)$. *I.e.* the vertices of H used to color S_i are exactly the colors of the set $\gamma(S_i)$.

Then, define the set $COL(C,\gamma)$ as the set of H-colorings of $G[\cup C]$ with profile γ. We see that for every red-connected component C of G_k, the sets $COL(C,\gamma)$ for $\gamma:C\mapsto(2^{V_H}\setminus\{\emptyset\})$ form a partition of the set of the H-colorings of $G[\cup C]$.

The principle of the algorithm is to inductively maintain (from $k=n$ to 1) the knowledge of every $|COL(C,\gamma)|$ (stored in a tabular $\#col(C,\gamma)$) for each red-connected component C of G_k and $\gamma:C\mapsto(2^{V_H}\setminus\{\emptyset\})$. In this way, since $\{V_G\}$ is a red-connected component of G_1, we can obtain the number of H-colorings of $G[V_G]=G$ by computing

$$\sum_{T\in(2^{V_H}\setminus\{\emptyset\})}\#col(\{V_G\},V_G\mapsto T).$$

First, note that the red-connected components of G_n are the $\{u\}$ for $u\in V_G$ (since G_n has no red edge). For every $\gamma:u\mapsto\gamma(u)\in(2^{|V_H|}\setminus\{\emptyset\})$ we let $\#col(\{u\},\gamma)\leftarrow 0$ if $|\gamma(u)|\neq 1$ and $\#col(\{u\},\gamma)\leftarrow 1$ if $|\gamma(u)|=1$. Hence, we correctly store the value of $|COL(\{u\},\gamma)|$ in the tabular $\#col(\{u\},\gamma)$.

We explain how to maintain this invariant after the contraction from G_{k+1} to G_k (with $k\in[n-1]$). By definition of a contraction sequence, G_k is of the form $G_k=:G_{k+1}/(U,V)$ with U and V two different vertices of G_{k+1}.

Note that every red-connected component of G_k is also a red-connected component of G_{k+1}, except the red-connected component C containing UV. We only have to compute $|COL(C,\gamma)|$ for any $\gamma:C\mapsto 2^{V_H}\setminus\{\emptyset\}$, and to store it in the tabular $\#col(C,\gamma)$. Initialize the value of $\#col(C,\gamma)$ with 0.

Let $C =: \{S_1 \dots, S_{p-1}, S'_p\}$, with $S'_p := UV$, and $p := |C| \leq \mathbf{ctww}(G)$. Since every pair of red-connected vertices in G_{k+1} (that contains neither U nor V) are red-connected in G_k, C must be of the form

$$C := (C_1 \cup \dots \cup C_q \cup \{S'_p\}) \setminus \{S_p, S_{p+1}\},$$

with $S_p := U$ and $S_{p+1} := V$ and $C_1 \cup \dots \cup C_q = \{S_1, \dots, S_{p-1}, S_p, S_{p+1}\}$,[10] and where $C_1, \dots, C_q$ (with $q > 0$) are red-connected components of G_{k+1} whose union contains both $S_p = U$ and $S_{p+1} = V$. Notice that each S_i (for $i \in [p+1]$) belongs to a unique $C_{j(i)}$ with $j(i) \in [q]$. These notions are illustrated in Figure 2.

The algorithm iterates over every family $(\gamma_j : C_j \mapsto (2^{V_H} \setminus \{\emptyset\}))_{1 \leq j \leq q}$. Let $\gamma = \gamma_1 \cup \dots \cup \gamma_q$ be the profile of C that maps every S_i (with $i \in [p-1]$) to $\gamma_{j(i)}(S_i)$, and that maps $S'_p = UV = S_p \cup S_{p+1}$ to $\gamma_{j(p)}(S_p) \cup \gamma_{j(p+1)}(S_{p+1})$. The algorithm checks if there exists a $(i, i') \in [p]^2$ with $i \neq i'$, a black edge between S_i and $S_{i'}$ in G_{k+1}, and $(\gamma(S_i) \times \gamma(S_{i'})) \subseteq E_H$, in time $O(p^2)$. If so, we increment $\#col(C, \gamma)$ by $\prod_{j=1}^{q} \#col(C_j, \gamma_j)$. Otherwise, we move to the next family $(\gamma_j)_{1 \leq j \leq q}$.

Soundness: For $(\gamma_j : C_j \mapsto (2^{V_H} \setminus \{\emptyset\}))_{1 \leq j \leq q}$, we denote by $COL(C, \gamma_1, \dots, \gamma_q)$ the sets of H-colorings f of C such that for all $j \in [q]$ the profile of $f|_{C_j}$ is γ_j. The algorithm is correct because, for each profile $\gamma : C \mapsto 2^{V_H} \setminus \{\emptyset\}$ of C, $COL(C, \gamma)$ is the disjointed union, for $(\gamma_1, \dots, \gamma_q)$ with $\gamma = \gamma_1 \cup \dots \cup \gamma_q$, of the $COL(C, \gamma_1, \dots, \gamma_q)$.

We only need to compute $|COL(C, \gamma_1, \dots, \gamma_q)|$, which can be derived by Claim 1. We then store the sum over $(\gamma_1, \dots, \gamma_q)$ such that $\gamma = \gamma_1 \cup \dots \cup \gamma_q$ in $\#col(C, \gamma)$. In Claim 1, we say that $(\gamma_1, \dots, \gamma_q)$ is *feasible* if for all $(i, i') \in [p]^2$ such that $(S_i, S_{i'})$ is a black edge of $G_{k+1}, (\gamma_{j(i)}(S_i) \times \gamma_{j(i')}(S_{i'})) \subseteq E_H$.

Claim 1. *We have for all $(\gamma_1, \dots, \gamma_q)$ that:*

1. *If $(\gamma_1, \dots, \gamma_q)$ is not feasible, then $COL(C, \gamma_1, \dots, \gamma_q) = \emptyset$.*
2. *If $(\gamma_1, \dots, \gamma_q)$ is feasible, then a function $f : \cup C \mapsto V_H$ belongs to $COL(C, \gamma_1, \dots, \gamma_q)$ if and only if, for all $j \in [q]$, f restricted to C_j (denoted by f_j) belongs to $COL(C_j, \gamma_j)$.*

Proof. We treat the two cases separately. First, we assume that $(\gamma_1, \dots, \gamma_q)$ is not feasible: there exists $(i, i') \in [p]^2$ such that $(S_i, S_{i'})$ is a black edge of G_{k+1} and $(\gamma_{j(i)}(S_i) \times \gamma_{j(i')}(S_{i'})) \setminus E_H \neq \emptyset$ and, for the sake of contradiction, suppose that there is $f \in COL(C, \gamma_1, \dots, \gamma_q)$. Take $(v_i, v_{i'}) \in (\gamma_{j(i)}(S_i) \times \gamma_{j(i')}(S_{i'})) \setminus E_H$. By definition of a profile, there exists $(u_i, u_{i'}) \in S_i \times S_{i'}$ with $f(u_i) = v_i$ and $f(u_{i'}) = v_{i'}$. Then, since there exists a black edge between S_i and $S_{i'}$ in G_{k+1}, this means by Property 1 that $(u_i, u_{i'}) \in E_G$. But $(f(u_i), f(u_{i'})) = (v_i, v_{i'}) \notin E_H$, so f is not an H-coloring, which contradicts the definition of f.

Second, we assume that $(\gamma_1, \dots, \gamma_q)$ is feasible. To prove necessity, notice that the restriction of a partial H-coloring is also a partial H-coloring, and by definition of $COL(C, \gamma_1, \dots, \gamma_q)$, if $f \in COL(C, \gamma_1, \dots, \gamma_q)$, then $f_j \in COL(C_j, \gamma_j)$.

[10] Note that $UV = S'_p = S_p \cup S_{p+1}$.

To prove sufficiency, assume that $f : \cup C \mapsto V_H$ is such that for all $j \in [q], f_j \in COL(C_j, \gamma_j)$. Then, provided that f is an H-coloring of $G[\cup C]$, $f \in COL(C, \gamma_1, \ldots, \gamma_q)$. Hence, we only have to prove that f is an H-coloring. So let $(u, u') \in E_G$. We prove that $(f(u), f(u')) \in E_H$. Observe that there exist S_i and $S_{i'}$ (with $(i, i') \in [p]^2$) such that $u \in S_i$ and $v \in S_{i'}$. If S_i and $S_{i'}$ are in the same red-connected component C_j (with $j \in [q]$) of G_{k+1}, then $(f(u), f(u')) = (f_j(u), f_j(u')) \in E_H$ because f_j is an H-coloring. Otherwise, $(S_i, S_{i'})$ is not a red edge of G_{k+1}, and since $(u, u') \in E_G$ and $(u, u') \in S_i \times S_{i'}$, it follows that so $(S_i, S_{i'})$ is a black edge of G_{k+1} by Property 1. By assumption of feasibility, $(\gamma_{j(i)}(S_i) \times \gamma_{j(i')}(S_{i'})) \subseteq E_H$ and, by definition of a profile, $(f(u), f(u')) = (f_{j(i)}(u), f_{j(i')}(u')) \in \gamma_{j(i)}(S_i) \times \gamma_{j(i')}(S_{i'}) \subseteq E_H$. The latter shows that f is indeed an H-coloring. □

From Claim 1 it follows that choosing an f in $COL(C, \gamma_1, \ldots, \gamma_q)$ is either impossible (if $(\gamma_1, \ldots, \gamma_q)$ is not feasible), or equivalent to choosing $f_j \in COL(C_j, \gamma_j)$ for all $j \in [q]$ (in case of feasibility), which is why we add either 0 or $\prod_{j=1}^{q} \#col(C_j, \gamma_j)$ when treating the part of $\#col(C, \gamma)$, relatively to the feasibility of the family $(\gamma_1, \ldots, \gamma_q)$.

Complexity: To treat the red-connected component C, the only non-polynomial part is to iterate over every family $(\gamma_1, \ldots, \gamma_q)$, which represents

$$\prod_{j=1}^{q} (2^{|V_H|} - 1)^{|C_j|} = (2^{|V_H|} - 1)^{|C|+1} \leq (2^{|V_H|} - 1)^{\mathbf{ctww}(G)+1}$$

families to treat (recall that for all $j \in [q]$, γ_j is a non-empty subset of C_j). □

If one only wishes to solve H-Coloring rather than the counting problem, the algorithm by Ganian et al. [5] which runs in $O^*(s(H)^{\mathbf{cw}(G)})$ for a graph parameter s, is strictly more efficient. The parameter $s(H)$ counts the number of different possible non-empty common neighborhoods for a subset of vertices of H. Indeed, for any graph H, its structural parameter $s(H)$ is bounded by $2^{|V_H|} - 2$ [5] (the equality happens if and only if H is a clique), and as we have proven in Proposition 4, for any graph G, $\mathbf{cw}(G) \leq \mathbf{ctww}(G) + 1$. However, it appears to be difficult to extend this algorithm to the counting problem since the sets stored as invariants in the algorithm do not necessarily represent disjoint subsets of partial coloring. This is acceptable if one only wants to determine the existence of a total coloring (as long as every coloring is represented at least once), but it causes issues when counting the number of colorings.

4.2 Fine-grained complexity

We now consider the dual problem of solving $\#H$-Coloring when H has bounded component twin-width. We therefore use an optimal contraction sequence of the template H instead of the input G, and obtain a fine-grained algorithm for $\#H$-Coloring which runs in $O^*((\mathbf{ctww}(H) + 2)^n)$ time.

Theorem 16. *$\#H$-Coloring is solvable in time $O^*((\mathbf{ctww}(H) + 2)^{|V_G|})$.*

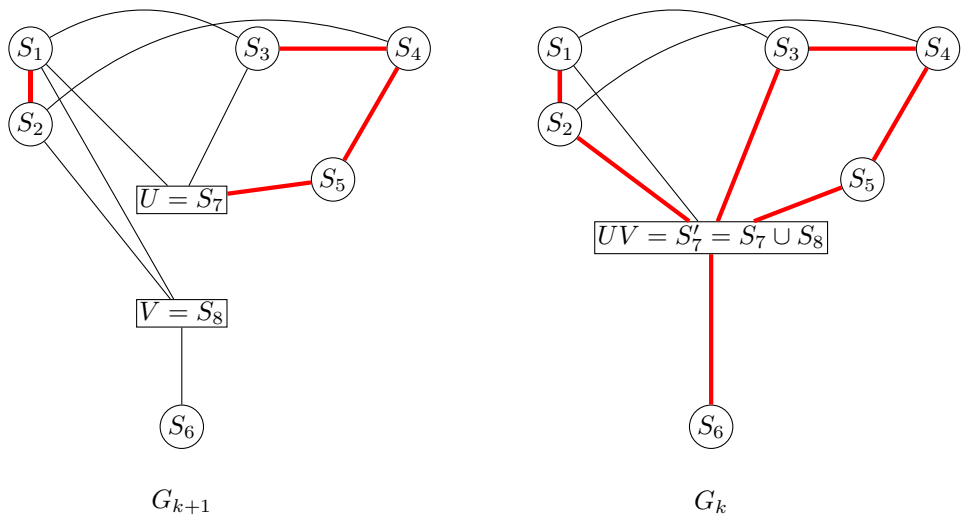

Fig. 2 An example where contracting $U = S_7$ and $V = S_8$ causes $j = 4$ different red-connected components to merge into a red-connected component of size $p = 7$. With the notations of this proof, we could have $C_1 = \{S_1, S_2\}, C_2 = \{S_3, S_4, S_5, S_7\}, C_3 = \{S_6\}$ and $C_4 = \{S_8\}$. For instance, $j(1) = j(2) = 1, j(3) = j(4) = j(5) = j(7) = 2, j(6) = 3$ and $j(8) = 4$.

Proof. Consider an optimal contraction sequence $(H_m, \ldots, H_1)$ of H, with $m := |V_H|$. Note that as H is part of the template and not part of the input, an optimal contraction sequence can be precomputed (for instance by exhaustive search). We give an algorithm similar to that described in the proof of Theorem 11, except that we define profiles for red-connected component of each H_k, with $k \in [m]$.

Let $C = \{T_1, \ldots, T_p\}$ be a red connected component of H_k and let $\gamma = (S_1, \ldots, S_p)$ be a p-tuple of pairwise disjoint subsets of V_G. An H-coloring f of $G[S_1 \cup \ldots, \cup S_p]$ is said to have *C-profile* γ if for each $i \in [p], f(S_i) \subseteq T_i$. Denote by $COL(\gamma, C)$ the set of partial H-colorings of G (*i.e.,* an H-Coloring of an induced subgraph) with C-profile γ. It is easy to compute the $|COL(\gamma, C)|$ for a red-connected component C of H_m (since H_m has no edge) and $\gamma = (S)$ with $S \subseteq V_G$, since C is of the form $C = \{v\}$ with $v \in V_H$. We have $|COL((S), \{v\})| = 1$ if $S^2 \cap E_G = \emptyset$, and $|COL((S), \{v\})| = 0$, otherwise.

As in the proof of Theorem 11, for $k \in [m-1]$ the only red-connected component of $H_k = H_{k+1}/(U, V)$ that is not a red-connected component of H_{k+1}, is the red-connected component $C = \{T_1, \ldots, T_{p-1}, T'_p\}$ that contains $T'_p = UV$ (the vertex obtained by contraction of $T_p = U$ and $T_{p+1} = V$ in H_{k+1}). Hence, C is of the form

$$C = (C_1 \cup \cdots \cup C_q \cup \{T'_p\}) \setminus \{T_p, T_{p+1}\},$$

with $C_1 \cup \cdots \cup C_q = \{T_1, \ldots, T_{p-1}, T_p, T_{p+1}\}$, where $C_1, \ldots, C_q$ are the red-connected components of H_{k+1} whose union contains $T_p = U$ and $T_{p+1} = V$. Again, each T_i belongs to a unique $C_{j(i)}$ with $j(i) \in [q]$.

Then, as in the proof of Theorem 11, for all families of disjoint subsets of V_G and $\gamma = (S_1, \ldots, S_{p-1}, S'_p)$, we can compute the value of $|COL(\gamma, C)|$. Indeed, as in the proof of Theorem 11, it is the sum for every family $(\gamma_j)_{1 \leq j \leq q}$ that defines the profile γ (*i.e.*, each γ_j is a family of pairwise disjoint subsets of V_G, and S'_p is of the form $S'_p = S_p \cup S_{p+1}$ with $S_p \cap S_{p+1} = \emptyset$ and $\forall \ell \in [q]$, $\gamma_\ell = (S_i)_{i \in j^{-1}(\{\ell\})}$[11]) of the value

1. $\prod_{j=1}^{q} |COL(\gamma_j, C_j)|$ if $(\gamma_1, \ldots, \gamma_q)$ is feasible,
2. 0 otherwise.

Here we say that $(\gamma_1, \ldots, \gamma_q)$ is *feasible* if for every $(i, i') \in [p]^2$ with $j(i) \neq j(i')$ and for every edge $(u_i, u_{i'})$ of G with $u_i \in S_i$ and $u_{i'} \in S_{i'}$, there is a black edge between T_i and $T_{i'}$ in H_{k+1},

The complexity of computing $|COL(\gamma, C)|$ for every γ is $(\mathbf{ctww}(H) + 2)^{|V_G|}$, since exploring every family $(\gamma_j)_{1 \leq j \leq q}$ containing only pairwise disjoint subsets of $|V_G|$ requires to explore $(\sum_{j=1}^{q} |C_j| + 1)^{|V_G|}$ families (any vertex of G can be mapped to a unique element in $\{T_1, T_2, \ldots, T_{p+1}\}$ or none of them), which makes $(|C| + 2)^n \leq (\mathbf{ctww}(H) + 2)^n$ possibilities. Since V_H is a red connected component of H_1, we obtain the number of such H-colorings of G in time $O^*((\mathbf{ctww}(H) + 2)^{|V_G|})$, and it is equal to $|COL(\{V_G\}, \{V_H\})|$. □

We again remark that, by Proposition 5, $\mathbf{ctww}(H) + 2 \leq \mathbf{lcw}(H) + 2$ and $\mathbf{ctww}(H) + 2 \leq 2\mathbf{cw}(H) + 1$ for any graph H. Therefore, the algorithm in the proof of Theorem 16 is always at least as fast as the clique-width approach by Wahlström [6], and as remarked in Section 1, it is strictly faster for e.g. cographs with edges and cycles of length ≥ 7, and distance hereditary graphs that are not cographs by Remark 3.

5 Conclusion and Future Research

In this article we explored component twin-width in the context of $\#H$-Coloring problems. We improved the bounds of the functional equivalence between component twin-width and clique-width from the (doubly) exponential $\mathbf{cw} \leq 2^{2^{\mathbf{ctww}}}$ and $\mathbf{ctww} \leq 2^{\mathbf{cw}+1}$ to the linear

$$\mathbf{cw} \leq \mathbf{ctww} + 1 \leq 2\mathbf{cw}.$$

In particular, this entails a single-exponential FPT algorithm for H-Coloring parameterized by component twin-width. From these linear bounds derives an approximation algorithm with exponential ratio, that can even be improved by a direct comparison with rank-width. We then demonstrated that our constructive proof technique could be extended to related parameters, and proved a quadratic bound between total twin-width and linear clique-width.

Finally, we turned to algorithmic applications, and constructed two algorithms for solving $\#H$-Coloring. The first uses a given optimal contraction sequence of the input graph G to solve $\#H$-Coloring in FPT time parameterized by component twin-width. The second uses a contraction sequence of the template graph H and beats the

[11] In other words, γ_ℓ is the tuple of the S_i where $i \in [p+1]$ is such that T_i belongs to the component C_ℓ.

clique-width approach for solving $\#H$-Coloring (with respect to $|V_G|$). Let us now discuss some topics for future research.

Tightness of the bounds. Even though the bound $\mathbf{cw} \leq \mathbf{ctww} + 1$ given by Proposition 4 is tight for any cograph with at least 1 edge, we do not currently know if this bound can be improved for graphs with greater clique-width or component twin-width. Moreover, it would be interesting to determine whether the bound $\mathbf{ctww} \leq 2\mathbf{cw} - 1$ given by Proposition 5 is tight. In particular, we believe that identifying classes of graphs, such as distance-hereditary graphs, for which a similar reasoning to the one presented in Remark 3 applies, constitutes a promising direction for future research. The same remark on tightness holds for the bounds between component twin-width and rank-width given by Theorem 9. It would be interesting to study the tightness of the bound $\mathbf{tww} \leq 2\mathbf{cw} - 2$ (where $\mathbf{tww}$ designs the twin-width), which is a direct consequence of Proposition 5. Also, since Propositions 13 and 14 provide very tight bounds, it is natural to ask for the characterization of the classes of graphs where each bound is attained.

Lower bounds on complexity. The algorithms relying on clique-width to solve H-Coloring by [5] in $O^*(s(H)^{\mathbf{cw}(G)})$ time are known to be optimal under the SETH. We have a similar optimality result for treewidth ($\mathbf{tw}$), with an algorithm solving H-Coloring in time $|V_H|^{\mathbf{tw}(G)}$, and the existence of a $(|V_H| - \varepsilon)^{\mathbf{tw}(G)}$ algorithm with $\varepsilon > 0$ being ruled out under SETH. A natural research direction is then to optimize the running time of the algorithm of Theorem 15, possibly by making use of $s(H)$, and prove a similar lower bound.

Extensions. Instead of solving $\#H$-Coloring the results of Section 4 can be extended to arbitrary binary constraints (*binary constraint satisfaction problems*, Bcsps). The notion of component twin-width indeed generalizes naturally to both instances and templates of a Bcsp. A natural continuation is then to investigate *infinite-domain* Bcsp*s* which are frequently used to model problems of interest in qualitative temporal and spatial reasoning. Here, there are only a handful of results using the much weaker treewidth parameter [46], so an FPT algorithm using component twin-width or clique-width would be a great generalization. Additionally, one may note that the algorithms detailed in Section 4 can be adapted to solve a "cost" version of $\#H$-Coloring: given a weight matrix C, the cost of a homomorphism f is $\sum_{u \in V_G} C(u, f(u))$, and we want to find a homomorphism of minimal cost. Can this be extended to other types of generalized problems?

References

[1] Formanowicz, P., Tanaś, K.: A survey of graph coloring-its types, methods and applications. Foundations of Computing and Decision Sciences **37**(3), 223–238 (2012)

[2] Dyer, M., Greenhill, C.: The complexity of counting graph homomorphisms. Random Structures & Algorithms **17**(3-4), 260–289 (2000)

[3] Fomin, F.V., Kratsch, D.: Exact Exponential Algorithms. Texts in Theoretical

Computer Science. An EATCS Series. Springer, Berlin, Heidelberg (2010)

[4] Fomin, F.V., Golovnev, A., Kulikov, A.S., Mihajlin, I.: Lower bounds for the graph homomorphism problem. In: Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP-2015). Lecture Notes in Computer Science, vol. 9134, pp. 481–493 (2015)

[5] Ganian, R., Hamm, T., Korchemna, V., Okrasa, K., Simonov, K.: The fine-grained complexity of graph homomorphism parameterized by clique-width. In: Bojanczyk, M., Merelli, E., Woodruff, D.P. (eds.) Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP-2022). LIPIcs, vol. 229, pp. 66–16620 (2022)

[6] Wahlström, M.: New plain-exponential time classes for graph homomorphism. Theory of Computing Systems **49**(2), 273–282 (2011)

[7] Bulatov, A.A., Dadsetan, A.: Counting homomorphisms in plain exponential time. In: Czumaj, A., Dawar, A., Merelli, E. (eds.) Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP-2020). LIPIcs, vol. 168, pp. 21–12118 (2020)

[8] Bergé, P., Bonnet, É., Déprés, H.: Deciding twin-width at most 4 is NP-complete. In: Bojanczyk, M., Merelli, E., Woodruff, D.P. (eds.) Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP-2022). LIPIcs, vol. 229, pp. 18–11820 (2022)

[9] Bonnet, E., Chakraborty, D., Kim, E.J., Köhler, N., Lopes, R., Thomassé, S.: Twin-Width VIII: Delineation and Win-Wins. In: Dell, H., Nederlof, J. (eds.) Proceedings of the 17th International Symposium on Parameterized and Exact Computation (IPEC-2022). Leibniz International Proceedings in Informatics (LIPIcs), vol. 249, pp. 9–1918 (2022)

[10] Bonnet, É., Déprés, H.: Twin-width can be exponential in treewidth. Journal of Combinatorial Theory, Series B **161**, 1–14 (2023)

[11] Bonnet, E., Geniet, C., Kim, E.J., Thomassé, S., Watrigant, R.: Twin-width III: Max Independent Set, Min Dominating Set, and Coloring. In: Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP-2021). Leibniz International Proceedings in Informatics (LIPIcs), vol. 198, pp. 35–13520 (2021)

[12] Bonnet, É., Geniet, C., Kim, E.J., Thomassé, S., Watrigant, R.: Twin-width II: small classes. In: Proceedings of the 32nd ACM-SIAM Symposium on Discrete Algorithms (SODA-2021), pp. 1977–1996 (2021)

[13] Bonnet, É., Geniet, C., Tessera, R., Thomassé, S.: Twin-width VII: groups. CoRR **abs/2204.12330** (2022)

[14] Bonnet, É., Giocanti, U., Mendez, P., Simon, P., Thomassé, S., Toruńczyk, S.: Twin-width IV: ordered graphs and matrices. In: Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC-2022), pp. 924–937 (2022)

[15] Bonnet, É., Kim, E.J., Reinald, A., Thomassé, S.: Twin-width VI: the lens of contraction sequences. In: Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-2022), pp. 1036–1056 (2022)

[16] Bonnet, É., Kim, E.J., Reinald, A., Thomassé, S., Watrigant, R.: Twin-width and polynomial kernels. Algorithmica **84**, 1–38 (2022)

[17] Bonnet, É., Kwon, O.-j., Wood, D.R., et al.: Reduced bandwidth: a qualitative strengthening of twin-width in minor-closed classes (and beyond). arXiv preprint arXiv:2202.11858 **""** (2022)

[18] Bonnet, É., Nešetřil, J., Mendez, P.O., Siebertz, S., Thomassé, S.: Twin-width and permutations. CoRR **abs/2102.06880** (2021)

[19] Schidler, A., Szeider, S.: A SAT approach to twin-width. In: Phillips, C.A., Speckmann, B. (eds.) Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX-2022), pp. 67–77 (2022)

[20] Gajarský, J., Pilipczuk, M., Przybyszewski, W., Torunczyk, S.: Twin-width and types. In: Bojanczyk, M., Merelli, E., Woodruff, D.P. (eds.) Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP-2022). LIPIcs, vol. 229, pp. 123–112321 (2022)

[21] Ahn, J., Hendrey, K., Kim, D., Oum, S.: Bounds for the twin-width of graphs. SIAM Journal on Discrete Mathematics **36**(3), 2352–2366 (2022)

[22] Dreier, J., Gajarský, J., Jiang, Y., Mendez, P.O., Raymond, J.: Twin-width and generalized coloring numbers. Discrete Mathematics **345**(3), 112746 (2022)

[23] Balabán, J., Hliněný, P.: Twin-width is linear in the poset width. arXiv preprint arXiv:2106.15337 (2021)

[24] Gajarský, J., Pilipczuk, M., Toruńczyk, S.: Stable graphs of bounded twin-width. In: Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science, pp. 1–12 (2022)

[25] Ganian, R., Pokrývka, F., Schidler, A., Simonov, K., Szeider, S.: Weighted model counting with twin-width. arXiv preprint arXiv:2206.01706 (2022)

[26] Král', D., Pekárková, K., Štorgel, K.: Twin-width of graphs on surfaces. arXiv preprint arXiv:2307.05811 (2023)

[27] Pilipczuk, M., Sokołowski, M., Zych-Pawlewicz, A.: Compact representation for

matrices of bounded twin-width. arXiv preprint arXiv:2110.08106 (2021)

[28] Balabán, J., Hliněný, P., Jedelský, J.: Twin-width and transductions of proper k-mixed-thin graphs. Discrete Mathematics, 113876 (2024)

[29] Pilipczuk, M., Sokołowski, M.: Graphs of bounded twin-width are quasi-polynomially χ-bounded. Journal of Combinatorial Theory, Series B **161**, 382–406 (2023)

[30] Hlinený, P., Jedelský, J.: Twin-width of planar graphs is at most 8, and at most 6 when bipartite planar. In: Etessami, K., Feige, U., Puppis, G. (eds.) Proceedings of the 50th International Colloquium on Automata, Languages, and Programming (ICALP-2023). LIPIcs, vol. 261, pp. 75–17518 (2023)

[31] Haan, R., Szeider, S.: Parameterized complexity classes beyond para-NP. Journal of Computer and System Sciences **87**, 16–57 (2017)

[32] Bui-Xuan, B.-M., Telle, J.A., Vatshelle, M.: Boolean-width of graphs. Theoretical Computer Science **412**(39), 5187–5204 (2011)

[33] Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms, 1st edn. Springer, Berlin, Heidelberg (2015)

[34] Bodlaender, H.L., Groenland, C., Jacob, H., Jaffke, L., Lima, P.T.: XNLP-completeness for parameterized problems on graphs with a linear structure. In: Dell, H., Nederlof, J. (eds.) Proceedings of the 17th International Symposium on Parameterized and Exact Computation (IPEC-2022). LIPIcs, vol. 249, pp. 8–1818 (2022)

[35] Fomin, F.V., Golovach, P.A., Lokshtanov, D., Saurabh, S., Zehavi, M.: Clique-width III: hamiltonian cycle and the odd case of graph coloring. ACM Transactions on Algorithms **15**(1), 9–1927 (2019)

[36] Golumbic, M.C., Rotics, U.: On the clique-width of some perfect graph classes. International Journal of Foundations of Computer Science **11**(03), 423–443 (2000)

[37] Flum, J., Grohe, M.: Parameterized Complexity Theory. Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin, Heidelberg (2006)

[38] Courcelle, B., Mosbah, M.: Monadic second-order evaluations on tree-decomposable graphs. Theoretical Computer Science **109**(1), 49–82 (1993)

[39] Corneil, D.G., Perl, Y., Stewart, L.K.: A linear recognition algorithm for cographs. SIAM Journal on Computing **14**(4), 926–934 (1985)

[40] Courcelle, B., Olariu, S.: Upper bounds to the clique width of graphs. Discrete

Applied Mathematics **101**(1-3), 77–114 (2000)

[41] Bonnet, É., Kim, E.J., Thomassé, S., Watrigant, R.: Twin-width I: tractable FO model checking. In: Proceedings of the 61st Annual Symposium on Foundations of Computer Science (FOCS-2020), pp. 601–612 (2020)

[42] Oum, S.-i.: Graphs of Bounded Rank-width. Princeton University, Princeton, New Jersey (2005)

[43] Gurski, F., Wanke, E.: On the relationship between nlc-width and linear nlc-width. Theoretical Computer Science **347**(1-2), 76–89 (2005)

[44] Oum, S.-i.: Approximating rank-width and clique-width quickly. ACM Transactions on Algorithms **5**(1), 1–20 (2008)

[45] Oum, S.-i., Seymour, P.: Approximating clique-width and branch-width. Journal of Combinatorial Theory, Series B **96**(4), 514–528 (2006)

[46] Dabrowski, K.K., Jonsson, P., Ordyniak, S., Osipov, G.: Solving infinite-domain CSPs using the patchwork property. In: Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-2021), pp. 3715–3723 (2021)

A Converting Contraction Sequences to *k*-expression and vice-versa

In this appendix we provide a visual example of the constructive proof of the bounds of Theorem 3.

Theorem 3. *For every graph G,* $\mathbf{cw}(G) \leq \mathbf{ctww}(G) + 1 \leq 2\mathbf{cw}(G)$.

We first illustrate the lefmost inequality in Section A.1, and then illustrate the rightmost part in Section A.2

A.1 From contraction sequences to *k*-expressions

We begin by recalling Proposition 4 .

Proposition 4. *For every graph G,* $\mathbf{cw}(G) \leq \mathbf{ctww}(G) + 1$.

As input the method takes a contraction sequence of the same graph that witnesses that its component twin-width is $\leq \kappa$, and and uses it do describe a $(\kappa+1)$-expression of a graph. Also note that the same construction illustrates Proposition 13 over the same graph (however, red-loops will not be represented).

Proposition 13. *For every graph G,* $\mathbf{lcw}(G) \leq \mathbf{tvtww}(G) + 1$.

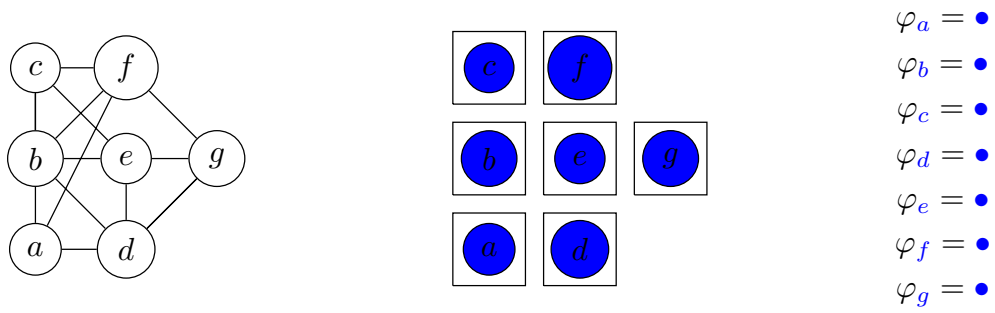

Fig. 3 Initial situation. All vertices are blue, but we can interchange labels within an expression if necessary.

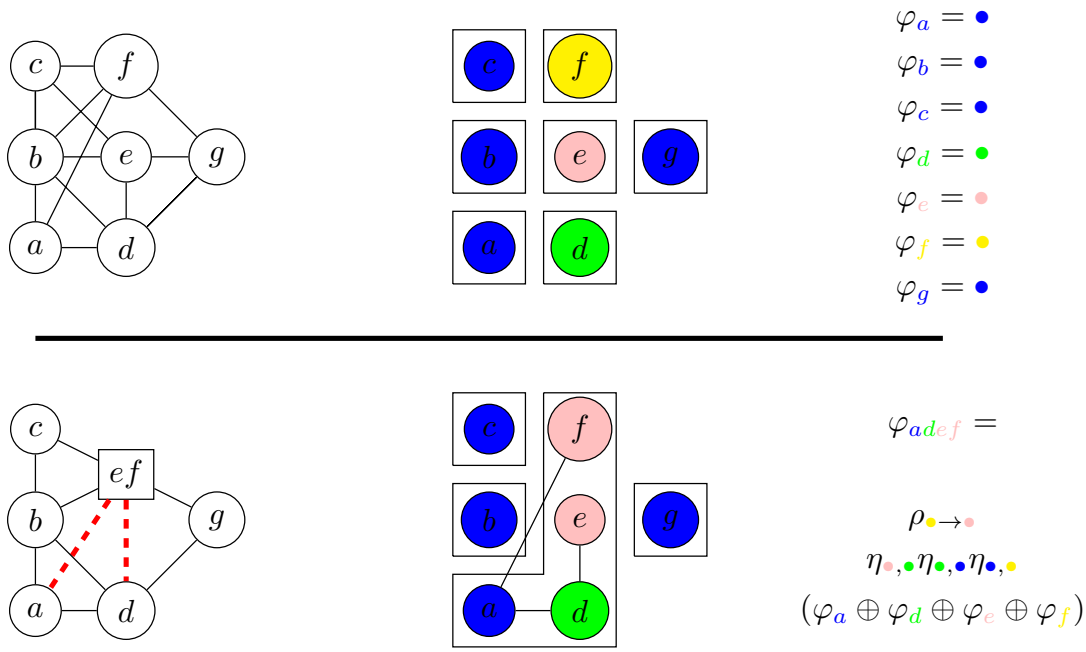

Fig. 4 We adapt the labels within components in anticipation of the contraction, perform disjoint unions, and construct the correct edges. Then, we set e and f to the same color.

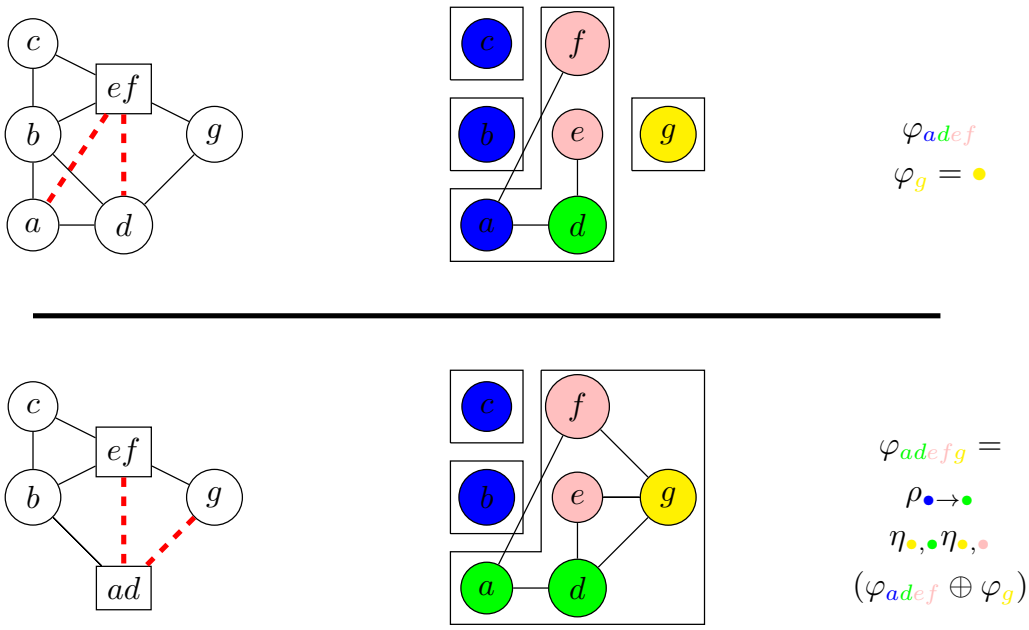

Fig. 5 Now, g joins the big red-connected component. Crucially, e and f "agree" on g.

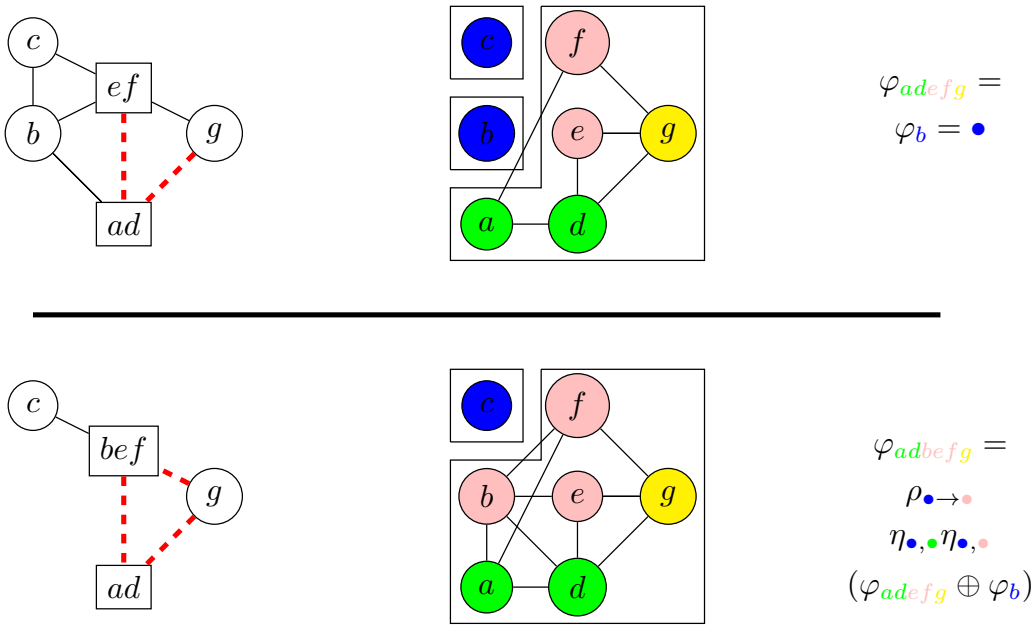

Fig. 6 b joins the "big" red-connected component

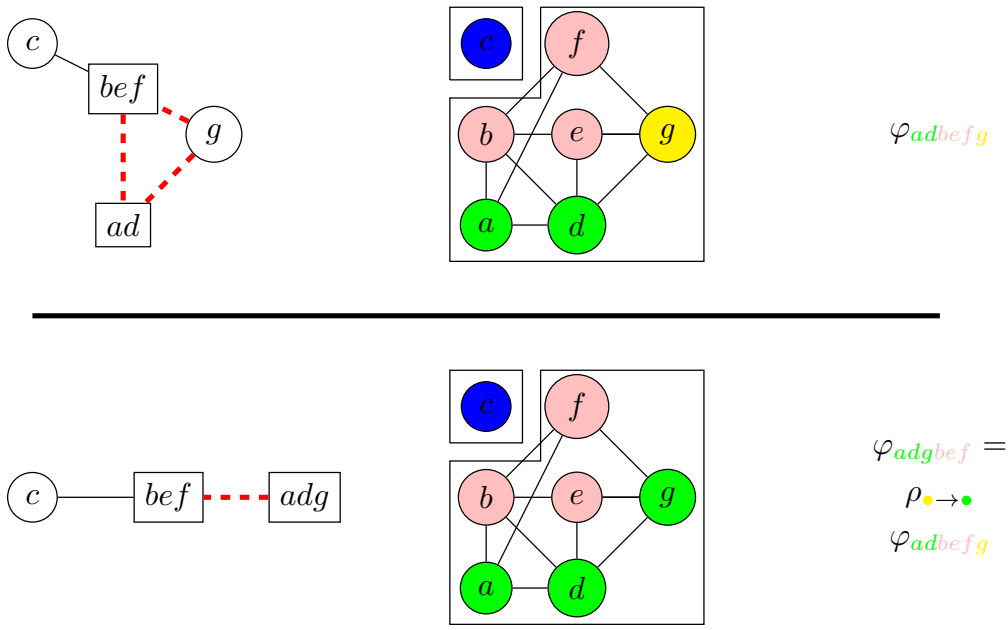

Fig. 7 The red-components are the same: only a relabelling happens.

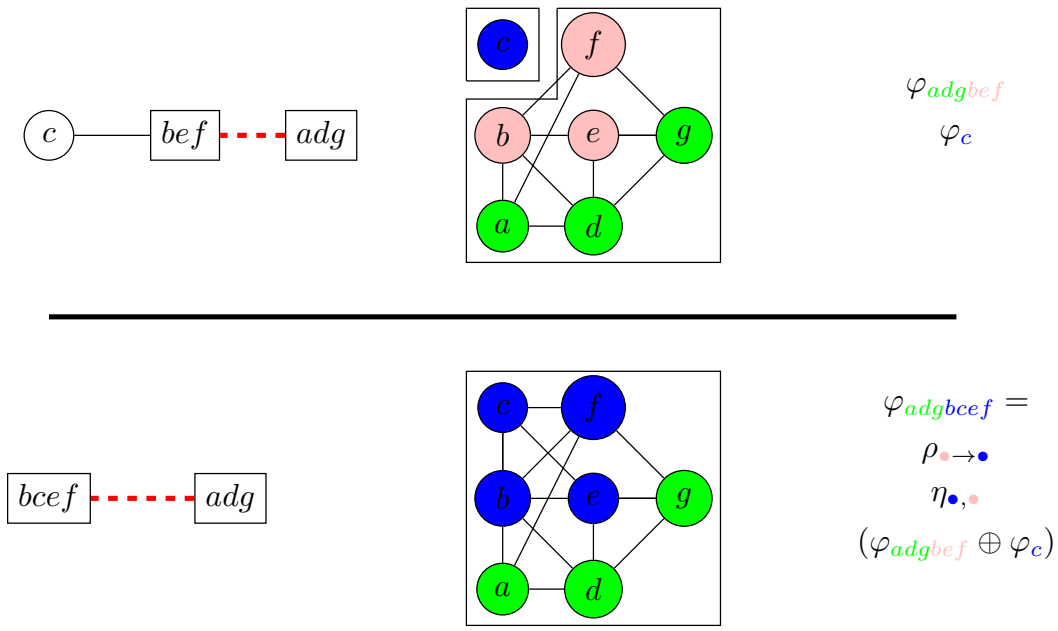

Fig. 8 Now c joins the "big component". We already have a 4-expression of the original graph.

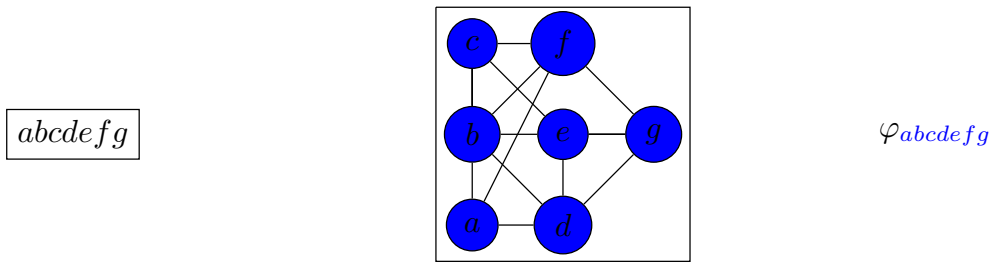

Fig. 9 Final situation.

A.2 From k-expressions to contraction sequences

We continue by illustrating an example of the application of the method described in Proposition 5 (establishing the rightmost part of the linear bounds of Theorem 3).

Proposition 5. *For every graph G, we have:*

(i) $\mathbf{ctww}(G) \leq 2\mathbf{cw}(G) - 1$, *and*
(ii) $\mathbf{ctww}(G) \leq \mathbf{lcw}(G)$.

We concentrate on illustrating (i), since (ii) is analogous. The method takes as an input a k-expression of a graph, and uses it to describe a contraction sequence of the same graph that witnesses that its component twin-width is $\leq 2k$.

This method progressively "collaps" the k-expression. A partition of the vertices of the original graph correspond naturally to every step of the collapse: two vertices are in the same subset of the partition if they have been collapsed together. Subsets of vertices that have been collapsed together are referred to as *parks*.

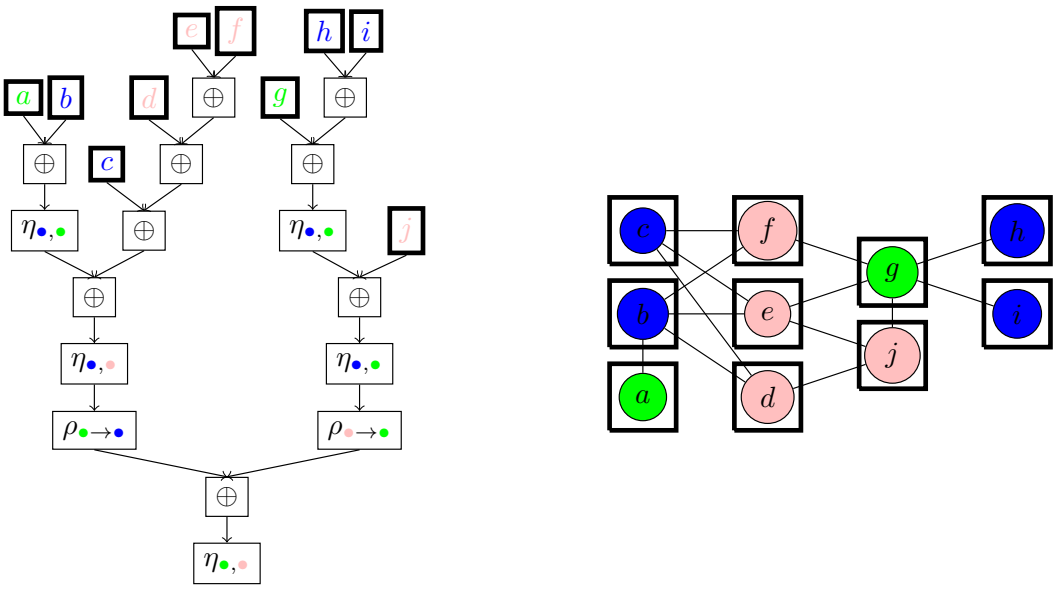

Fig. 10 We represent the vertices with their current label.

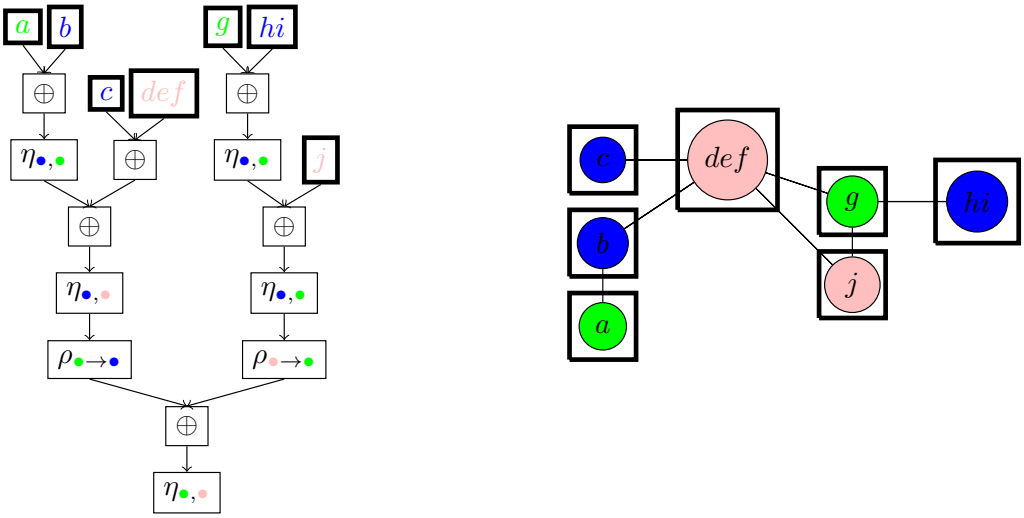

Fig. 11 d, e and f are introduced together with the same label: they are twins. We can contract them.

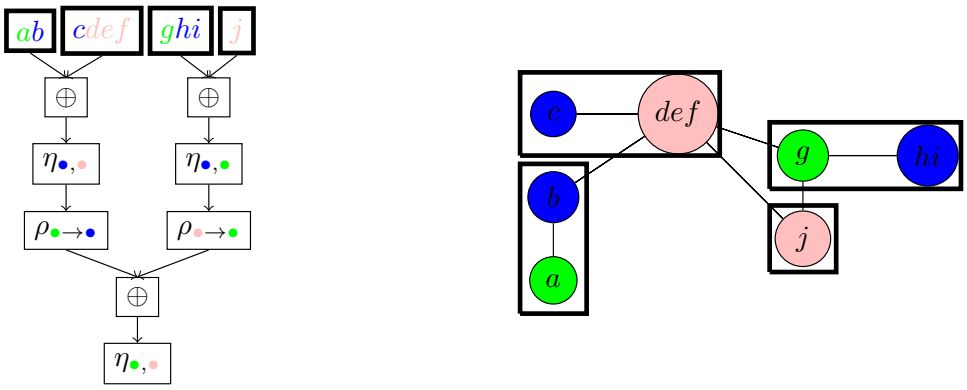

Fig. 12 We collapse the k-expression and merge the "parks" accordingly.

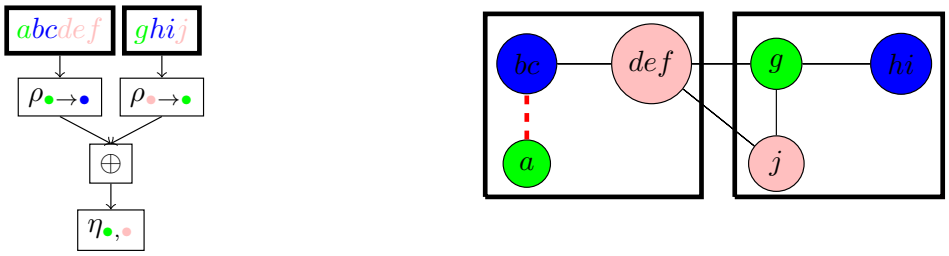

Fig. 13 We merge vertices with the same label in the same park: the red-edges created are confined in the parks.

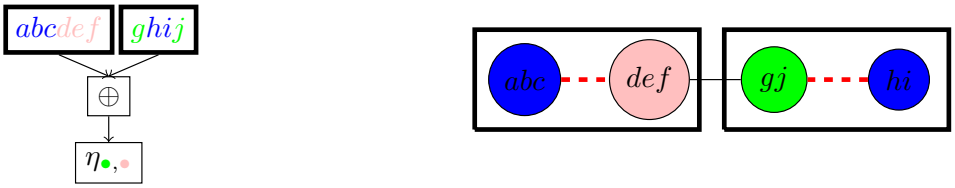

Fig. 14 After the next step, only one park will remain. We can finish the contraction sequence arbiltrarly.

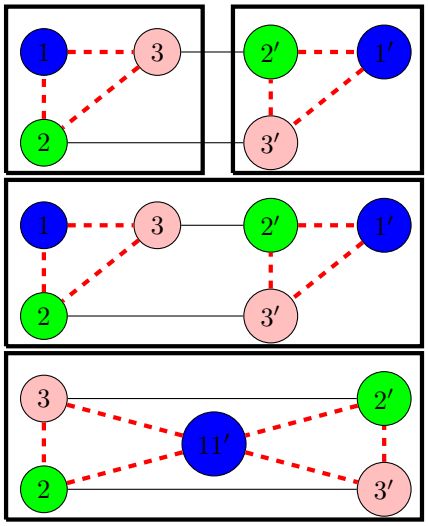

Fig. 15 Component twin-width in the worst case: at worst we merge two colorful parks (with $\mathbf{cw}(G)$ vertices), and the next contraction will create a red connected component of size $2\mathbf{cw}(G) - 1$.

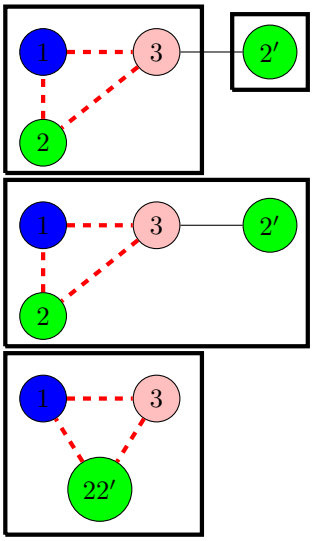

Fig. 16 If the expression is linear, the worst case component twin-width becomes $\mathbf{lcw}(G)$.