

Mim-Width is paraNP-complete

Benjamin Bergougnoux   

LIS, Aix-Marseille Université, France

Édouard Bonnet   

CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR 5668, Lyon, France

Julien Duron   

ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR 5668, Lyon, France

Abstract

We show that it is NP-hard to distinguish graphs of linear mim-width at most 1211 from graphs of sim-width at least 1216. This implies that MIM-WIDTH, SIM-WIDTH, ONE-SIDED MIM-WIDTH, and their linear counterparts are all paraNP-complete, i.e., NP-complete to compute even when upper bounded by a constant. A key intermediate problem that we introduce and show NP-complete, LINEAR DEGREE BALANCING, inputs an edge-weighted graph G and an integer τ , and asks whether $V(G)$ can be linearly ordered such that every vertex of G has weighted *backward* and *forward* degrees at most τ .

1 Introduction

While it was shown shortly after the inception of these parameters by Vatschelle in 2012 [14, 1] that MIM-WIDTH and LINEAR MIM-WIDTH are W[1]-hard [12, 13], whether a slice-wise polynomial (XP) algorithm¹ can compute (or approximate) the (linear) mim-width of an input graph has been raised as an open question repeatedly over the past twelve years [14, 13, 10, 9, 4, 3, 11, 2]. We give a negative answer to this question (at least for some too-good approximation factor), and similarly settle the parameterized complexity of the related sim-width and one-sided mim-width parameters, as well as their linear variants. Indeed we show that all these parameters are paraNP-complete to compute, i.e., NP-complete even when guaranteed to be upper bounded by a universal constant.

► **Theorem 1.** *MIM-WIDTH, SIM-WIDTH, ONE-SIDED MIM-WIDTH, LINEAR MIM-WIDTH, LINEAR SIM-WIDTH, and LINEAR ONE-SIDED MIM-WIDTH are paraNP-complete.*

We show Theorem 1 with a single reduction.

► **Theorem 2.** *There is a polynomial-time algorithm that takes an input φ of 4-OCC NOT-ALL-EQUAL 3-SAT and builds a graph G^* such that*

- *if φ is satisfiable, then G^* has linear mim-width at most 1211,*
- *if φ is unsatisfiable, then G^* has sim-width at least 1216.*

Theorem 2 indeed implies Theorem 1 as the linear mim-width upper bounds the other five parameters, while the sim-width lower bounds the other five parameters. Our reduction is naturally split into three parts, thereby going through two intermediate problems. The first intermediate problem may be of independent interest (perhaps especially so, its unweighted version), and we were somewhat surprised not to find it already defined in the literature. We call it LINEAR DEGREE BALANCING.

¹ i.e., for any fixed integer k , a polynomial-time algorithm (whose exponent may depend on k) that decides if the (linear) mim-width of the input graph is at most k .

LINEAR DEGREE BALANCING

Parameter: τ

Input: An edge-weighted n -vertex graph H and a non-negative integer τ .

Question: Is there a linear ordering $v_1 \prec v_2 \prec \dots \prec v_n$ of $V(H)$ such that every vertex v_i has weighted degree in $H[\{v_1, \dots, v_i\}]$ and in $H[\{v_i, \dots, v_n\}]$ at most τ ?

We call τ -balancing order of H a linear order over $V(H)$ witnessing that H is a positive instance of LINEAR DEGREE BALANCING.

The three steps. Our reduction starts with a NOT-ALL-EQUAL 3-SAT (NAE 3-SAT for short) instance φ , and goes through an edge-weighted graph (H, ω) , a vertex-partitioned graph (G, \mathcal{P}) , and finally an instance G^* of MIM-WIDTH.

First we prove that LINEAR DEGREE BALANCING is NP-complete even when τ is a constant, and every edge weight is a positive integer. For our purpose, we in fact show something stronger. The first step is a polynomial-time reduction from 4-OCC NAE 3-SAT that maps satisfiable formulas to edge-weighted graphs admitting a τ -balancing order, and unsatisfiable formulas to negative instances of TREE DEGREE BALANCING, a tree variant of LINEAR DEGREE BALANCING, for the larger threshold of $\tau + \gamma$, where γ can grow linearly in τ .

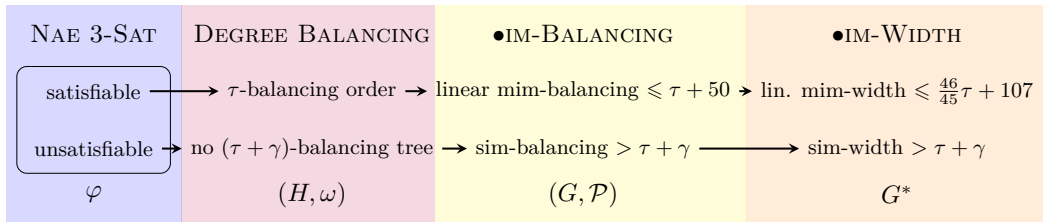
In TREE DEGREE BALANCING, the vertices of H are bijectively mapped to the nodes of a freely-chosen tree T such that for every node t of T and every edge e incident to t , the vertex of H mapped to t has weighted degree at most the given threshold in the cut of H defined by the two connected components of $T - e$. A formal definition is given in Section 3.5.

The second step turns the weighted degree into the maximum (semi-)induced matching at the expense of mapping *subsets* of vertices of G to nodes of T , in a way that the nodes of T jointly hold a prescribed partition \mathcal{P} of $V(G)$. In TREE MIM-BALANCING (resp. TREE SIM-BALANCING), for every edge e of T , the size of a maximum semi-induced (resp. induced) matching in the cut of G defined by e shall remain below the threshold. Their linear variants force T to be a path. See Section 4.1 for formal definitions.

The third step erases the differences between TREE MIM-BALANCING and MIM-WIDTH, and between their respective variants. Intuitively speaking:

- for each part of \mathcal{P} , the corresponding vertices of G^* can be gathered in their own subtree,
- T can be chosen ternary (i.e., every non leaf node has degree 3),
- only the leaves of T need hold a vertex of G^* .

Figure 1 summarizes these three steps.



■ **Figure 1** Visual summary of our reduction, split into its three steps.

We now outline each step.

Nae 3-Sat to Degree Balancing. We actually reduce from the positive variant of NAE 3-SAT, where no literal is negated. We design a gadget called *bottleneck sequence* that, given three disjoint sets $X, Y, Z \subset V(H)$, forces all vertices of Y to appear in the order after all the vertices of X , and before all the vertices of Z (or by symmetry after all the vertices of Z ,

and before all the vertices of X). Vertices of Y are in one-to-one correspondence with clauses of φ . Similarly, we have a vertex for each variable of φ . Each *variable* vertex is forced to be placed before X (where it represents being set to true), or after Z (where it represents being set to false). The weights are designed so that a *clause* vertex can tolerate two but not three of its *variable* vertices to be on the same side (before or after it); which exactly captures the semantic of a not-all-equal 3-clause.

The gap between *at most* τ and *at least* $\tau + \gamma + 1$ is obtained by carefully crafting ω . We also add a padding gadget to raise the minimum degree of H , in such a way that only two vertices have low-enough degree to be leaves of T . This forces T to be a path (the only tree with at most two leaves), thus LINEAR DEGREE BALANCING and TREE DEGREE BALANCING to coincide.

Degree Balancing to {Linear M, Tree S}im-Balancing. Every vertex u of H becomes an independent set $S(u)$ of G and a part of \mathcal{P} of size the sum of the weights of edges incident to u . Adjacencies in H become induced matchings in G , whereas non-adjacencies in H become bicliques in G (with some additional twist, see Figure 5). The density of G forces large induced matchings to be mainly incident to a single part $S(u)$. Thus, roughly speaking, the maximum induced matchings in G behave like the degree in H . As the parts $S(u)$ are independent sets, there is in effect no difference between TREE MIM-BALANCING and TREE SIM-BALANCING. The indifference between the tree or the linear variants is inherited from the previous reduction. The actual arguments incur a small additive loss (of 50) in the induced matching size, which is eventually outweighed by γ .

{Linear M, Tree S}im-Balancing to {Linear M, S}im-Width. We design a *part gadget* $\mathcal{G}(u)$ that simultaneously takes care of the three items above Figure 1. Essentially, every part $S(u)$ is transformed into the 1-subdivision P_u of a path on $|S(u)|$ vertices, then duplicated a large (but constant) number of times, concatenated into a single path, and every pair of vertices in different copies are linked by an edge whenever they do not correspond to the same vertex or neighboring vertices in P_u . On the one hand, this may only increase the linear mim-width (compared to the linear mim-balancing) by an additive constant. Following the “spine” of $\mathcal{G}(u)$, one gets a witness of low linear mim-width for G^* from a witness of low linear mim-balancing of (G, \mathcal{P}) .

On the other hand, the dense “path-like” structure of $\mathcal{G}(u)$ ensures that, in an optimal decomposition of G^* , its vertices may as well be placed in order at the leaves of a caterpillar. We thus devise a process that builds a witness of low sim-balancing for (G, \mathcal{P}) from a witness of low sim-width for G^* : We in turn identify an edge e of the branch decomposition of G^* that can support $V(\mathcal{G}(u))$, and in particular $S(u)$, without increasing the width. We then relocate the vertices of $S(u)$ at a vertex subdividing e . Eventually each set $S(u)$ is solidified at a single node of the tree, and we reach the desired witness for TREE SIM-BALANCING.

Remarks and perspectives. It can be noted that we had to develop completely new techniques. Indeed, the known W[1]-hardness [12, 13] relies on the difficulty of actually computing the value of a fixed cut, i.e., solving MAXIMUM INDUCED MATCHING. In some sense, the instances produced there are not difficult to solve (a best decomposition is, on the contrary, suggested by the reduction), but only to evaluate. In any case, as MAXIMUM INDUCED MATCHING is W[1]-hard but admits a straightforward XP algorithm, we could not use the same idea.

We believe that LINEAR DEGREE BALANCING could be explored for its own sake. We emphasize that our techniques could prove useful to show the paraNP-hardness of other parameters based on branch decompositions such as those recently introduced by Eiben et

al. [6], where the *cut function* can combine maximum (semi-)induced matchings, maximum (semi-)induced co-matchings, maximum half-graphs (or ladders). One would then mainly need to tune the gadgets of the second step (see Figure 5) to fit the particular cut function.

We notice that our reduction from 4-OCC NOT-ALL-EQUAL 3-SAT is linear: n -variables instances are mapped to $\Theta(n)$ -vertex graphs. Hence, unless the Exponential-Time Hypothesis (ETH) [7] fails,² no $2^{o(n)}$ -time algorithm can decide if the mim-width (or any of the five variants of mim-width) of an n -vertex graph is at most 1211. Indeed the absence of $2^{o(n)}$ -time algorithm for n -variable 4-OCC NOT-ALL-EQUAL 3-SAT (even POSITIVE 4-OCC NOT-ALL-EQUAL 3-SAT) under the ETH can be derived from the Sparsification Lemma [8] and classic reductions.

Our focus was to handle all the variants of mim-width at once. This made the reduction more technical and degraded the constant upper and lower bounds. Better bounds (than 1211) could be achieved if separately dealing with MIM-WIDTH or with LINEAR MIM-WIDTH. For example, the latter problem essentially only requires the first two steps of the reduction. Still, deciding if the (linear) mim-width of a graph is at most 1 (or any 1-digit constant) remains open. In addition, the question whether an XP $f(\text{OPT})$ -approximation algorithm for MIM-WIDTH (and its variants) exists, for some fixed function f and OPT being the optimum width, remains open.

2 Graph definitions and notation

For i and j two integers, we denote by $[i, j]$ the set of integers that are at least i and at most j . For every integer i , $[i]$ is a shorthand for $[1, i]$.

2.1 Standard graph theory

We denote by $V(G)$ and $E(G)$ the vertex and the edge set, respectively, of a graph G . If G is a graph and $S \subseteq V(G)$, we denote by $G[S]$ the subgraph of G induced by S , and use $G - S$ as a short-hand for $G[V(G) \setminus S]$. If $e \in E(G)$, we denote by $G - e$ the graph G deprived of edge e , but the endpoints of e remain. More generally, if $F \subseteq E(G)$, $G - F$ is the graph obtained from G by removing all the edges of F (but not their endpoints). For $X \subseteq V(G)$, we may denote by $E_G(X)$ the edge set of $G[X]$. We denote the open and closed neighborhoods of a vertex v in G by $N_G(v)$ and $N_G[v]$, respectively. For $S \subseteq V(G)$, we set $N_G(S) := \bigcup_{v \in S} N_G(v) \setminus S$ and $N_G[S] := N_G(S) \cup S$. In every notation with a graph subscript, we may omit it if the graph is clear from the context. A vertex set $S \subseteq V(G)$ *covers* an edge set $F \subseteq E(G)$ if every edge of F has at least one endpoint in S .

A *cut* of a graph G is a bipartition (A, B) of $V(G)$. The *cut-set* defined by a cut (A, B) , denoted by $E(A, B)$, is $\{uv \in E(G) \mid u \in A, v \in B\}$. We denote by $G[A, B]$ the bipartite subgraph of G with edge set $E(A, B)$. A *matching* is a set of edges that share no endpoints and an *induced matching* of G is a matching M such that every edge of G intersects at most one edge in M . If $A, B \subseteq V(G)$ are two disjoint vertex subsets of G , a *matching between A and B* is a matching where every edge has one endpoint in A and the other endpoint in B . An induced matching in $G[A, B]$ is called a *semi-induced* matching of G between A and B .

² the assumption that there is a $\lambda > 0$ such that n -variable 3-SAT cannot be solved in time $O(\lambda^n)$.

2.2 Mim-width and its variants

A *branch decomposition* or *tree layout* (or simply *layout*) of a graph G is a pair (T, f) where T is a ternary tree (i.e., every internal node of T has degree 3) and f is a bijection from $V(G)$ to the leaves of T . Given two disjoint sets $X, Y \subseteq V(G)$, we denote by $\text{mim}_G(X, Y)$ (resp. $\text{sim}_G(X, Y)$) the maximum number of edges in a semi-induced matching (resp. induced matching) of G between X and Y , and may refer to it as *mim-value* (resp. *sim-value*). An edge e of T *induces* or *defines* a cut (A_e, B_e) of G , where A_e and B_e are the preimages by f of the leaves in the two components of $T - e$.

The *mim-value* (resp. *sim-value*) of (A_e, B_e) is set as $\text{mim}_G(A_e, B_e)$ (resp. $\text{sim}_G(A_e, B_e)$). The *mim-value* (resp. *sim-value*) of the branch decomposition (T, f) is the maximum of $\text{mim}_G(A_e, B_e)$ (resp. $\text{sim}_G(A_e, B_e)$) taken over every edge e of T . Finally, the *mim-width* (resp. *sim-width*) of G is the minimum *mim-value* (resp. *sim-value*) taken over every branch decomposition (T, f) of G .

The *upper-induced matching number* of $X \subseteq V(G)$ is the maximum size of an induced matching of $G - E(V(G) \setminus X)$ between X and $V(G) \setminus X$. The *one-sided mim-width* is defined as above with the *omim-value* of cut (A_e, B_e) , $\text{omim}_G(A_e, B_e)$, defined as the minimum between the upper-induced matching numbers of A_e and of B_e .

The linear variants of these widths and values impose T to be a rooted *full* binary tree (i.e., every internal node has exactly two children) such that the internal nodes form a path.

3 Not-All-Equal 3-Sat to Degree Balancing

Given a graph H edge-weighted by a map $\omega : E(H) \rightarrow \mathbb{N}$, the *weight of a vertex* v of H is the sum of the weights of the edges incident to v . We say that a total order \prec on $V(H)$ is τ -*balancing*, for some non-negative integer τ , if for every vertex $v \in V(H)$ the *left weight* of v , $\sum_{u \in N(v), u \prec v} \omega(uv)$, and the *right weight* of v , $\sum_{u \in N(v), v \prec u} \omega(uv)$, are both at most τ , i.e.,

$$\Delta_{\prec}(v) := \max \left(\sum_{u \in N(v), u \prec v} \omega(uv), \sum_{u \in N(v), v \prec u} \omega(uv) \right) \leq \tau.$$

Constants τ, γ, λ . Henceforth we will use τ and γ as global natural constants. The reduction in this section will also use a constant positive integer λ . For the current section, we need that the following conditions hold.

$$\gamma < \lambda, \quad 3\gamma + 4 < \tau, \quad 2\lambda + \gamma < \tau, \quad 6\lambda \leq \tau. \quad (1)$$

We will not only prove that LINEAR DEGREE BALANCING is paraNP-hard but we will obtain a scalable additive gap. More specifically, we start by showing the following.

► **Theorem 3.** *It is NP-hard to distinguish graphs having a τ -balancing order from graphs having no $(\tau + \gamma)$ -balancing order.*

Eventually we will need that τ and γ are multiples of a constant integer a (which is defined and set to 45 in Section 5). This can simply be achieved by multiplying all edge weights of the forthcoming reduction by a . We will finally set $\tau := 24a = 1080$, $\lambda := 4a = 180$, and $\gamma := 3a = 135$. One can quickly check that these values do respect Equation (1).

3.1 First properties on balancing orders, and bottlenecks

Given a total order \prec on a graph H , we say that a vertex set S is *smaller* (resp. *larger*) than another vertex set U , denoted by $S \prec U$ (resp. $U \prec S$), if for all $s \in S, u \in U$ we have $s \prec u$ (resp. $u \prec s$). When a set S is neither larger nor smaller than a vertex u , we say that S *surrounds* u . We also say that u is *surrounded by* S . Note that if S surrounds two vertices u and v , it surrounds any vertex w with $u \prec w \prec v$.

We begin with a useful observation on the only possible τ -balancing order of a P_3 (i.e., 3-vertex path) with large total weight.

► **Lemma 4.** *For any integer t , and any edge-weighted graph (H, ω) containing a P_3 abc such that $\omega(ab) + \omega(bc) > t$. Then in any t -balancing order of H , $\{a, c\}$ surrounds b .*

Proof. TOPROVE 0 ◀

We also rely on the following observation, where the induced subgraph of an edge-weighted graph (H, ω) is an induced subgraph of H edge-weighted by the restriction of ω to its edge set.

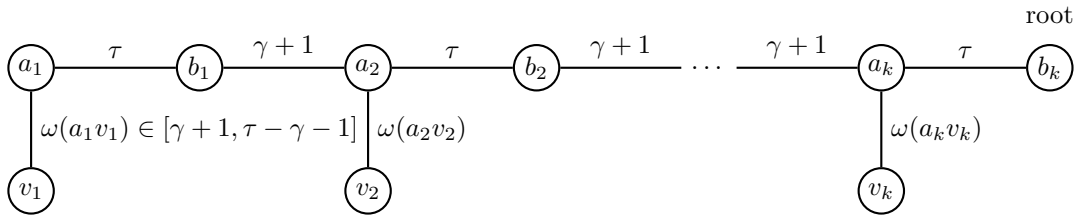
► **Observation 5.** *Every t -balancing order of (H, ω) is a t -balancing order of any induced subgraph of (H, ω) .*

Our main ingredient here is called *bottleneck*.

► **Definition 6.** *A (τ, γ) -bottleneck on terminals v_1, \dots, v_k is an edge-weighted caterpillar B defined as follows.*

1. *Let $P(B)$ be a $2k$ -vertex path, say $a_1b_1a_2b_2 \dots a_kb_k$, called spine of B and for every $i \in [k]$, we set $\omega(a_ib_i) := \tau$ and $\omega(b_ia_{i+1}) := \gamma + 1$.*
2. *We obtain B by adding to $P(B)$ a leaf v_i adjacent to a_i , satisfying $\gamma + 1 \leq \omega(v_ia_i) \leq \tau - \gamma - 1$ for every $i \in [k]$. This edge is called the attachment of v_i to B .*
3. *The caterpillar B is rooted in b_k .*

The vertex v_1 is called *first terminal* of B . A (τ, γ) -bottleneck is depicted in Figure 2.



■ **Figure 2** Illustration of a (τ, γ) -bottleneck.

A bottleneck ensures the following.

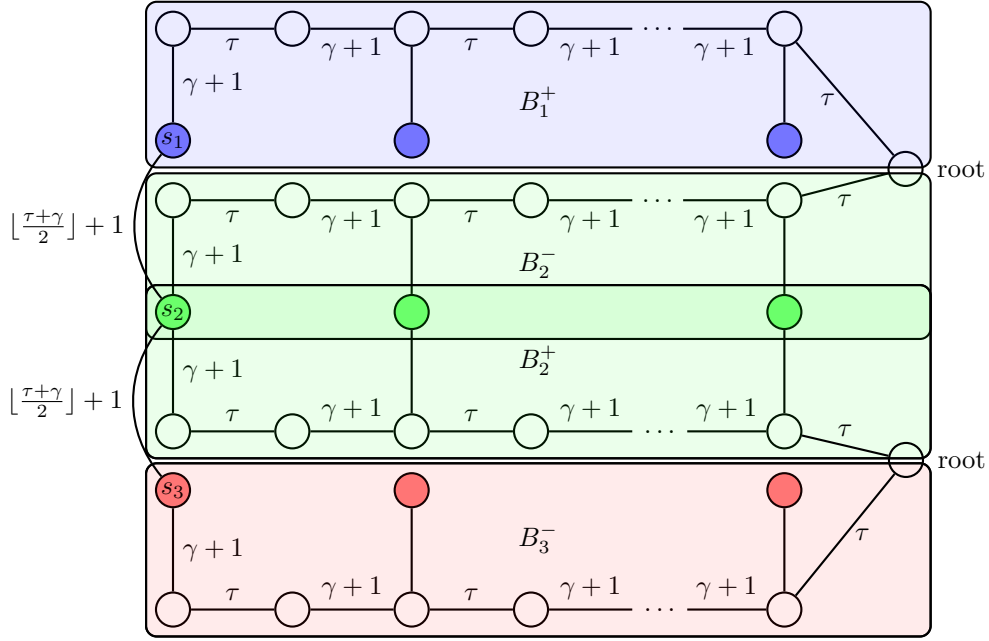
► **Lemma 7.** *Let \prec be a $(\tau + \gamma)$ -balancing order of a (τ, γ) -bottleneck B on terminals v_1, \dots, v_k . Using the notation of Definition 6, if $a_k \prec b_k$ then $a_1 \prec b_1 \prec a_2 \prec b_2 \prec \dots \prec a_k \prec b_k$, and $v_i \prec a_i$ for each $i \in [k]$. Hence symmetrically, if $b_k \prec a_k$ then $b_k \prec a_k \prec b_{k-1} \prec a_{k-1} \prec \dots \prec b_1 \prec a_1$, and $a_i \prec v_i$ for each $i \in [k]$.*

Proof. TOPROVE 1 ◀

Henceforth every bottleneck is a (τ, γ) -bottleneck. Thus we simply write *bottleneck*.

► **Definition 8.** Given three vertex sets S_1, S_2, S_3 , we call bottleneck sequence on S_1, S_2, S_3 an edge-weighted graph $B(S_1, S_2, S_3)$ obtained by adding

1. for every $i \in \{1, 2\}$, a bottleneck B_i^+ with terminals $S_i \cup \{s_i\}$ where s_i is the first terminal of B_i^+ , and the attachment of s_i is of weight $\gamma + 1$,
 2. for every $i \in \{2, 3\}$, a bottleneck B_i^- with terminals $S_i \cup \{s_i\}$ where s_i is the first terminal of B_i^- and the attachment of s_i is of weight $\gamma + 1$ such that
 3. for every $i \in \{1, 2\}$, the roots of B_i^+ and of B_{i+1}^- are identified as the same vertex, and
 4. for every $i \in \{2, 3\}$, an edge $s_i s_{i+1}$ of weight $\lfloor \frac{\tau + \gamma}{2} \rfloor + 1$,
- with s_1, s_2, s_3 three new vertices.



■ **Figure 3** Bottleneck sequence $B(S_1, S_2, S_3)$. Vertices of $S_1 \cup \{s_1\}, S_2 \cup \{s_2\}, S_3 \cup \{s_3\}$ are in red, green, and blue, respectively. As in Figure 2, every edge with an unspecified weight get one in the discrete interval $[\gamma + 1, \tau - \gamma - 1]$.

The next lemma yields the crucial property ensured by bottleneck sequences.

► **Lemma 9.** Any $(\tau + \gamma)$ -balancing order \prec on a bottleneck sequence $B(S_1, S_2, S_3)$ is such that $S_1 \prec S_2 \prec S_3$ or $S_3 \prec S_2 \prec S_1$.

Proof. TOPROVE 2 ◀

We conclude the section by defining τ -balancing orders for bottleneck sequences. A *direct order* \prec_{\rightarrow} of a bottleneck B with terminals v_1, \dots, v_k goes as follows:

$$\{v_1, \dots, v_k\} \prec_{\rightarrow} a_1 \prec_{\rightarrow} b_1 \prec_{\rightarrow} a_2 \prec_{\rightarrow} b_2 \prec_{\rightarrow} \dots \prec_{\rightarrow} a_k \prec_{\rightarrow} b_k,$$

where $a_1 b_2 \dots a_k b_k$ is the spine of B rooted in b_k . Note that the order induced by $\{v_1, \dots, v_k\}$ is not specified (and so a given bottleneck on k terminals admits $k!$ different direct orders). A *reverse order* of B , denoted by \prec_{\leftarrow} , is simply defined as the reverse order of a direct order \prec_{\rightarrow} .

A *direct order* $\prec_{\rightarrow}^{\text{seq}}$ of a bottleneck sequence $B(S_1, S_2, S_3)$ is a common (linear) extension of direct orders on B_1^+ and B_2^+ and reverse orders on B_2^- and B_3^- . Note that on any

bottleneck sequence, at least one direct order exists since the direct and reverse orders constrain disjoint vertex sets. In particular we have $S_1 \prec_{\rightarrow}^{\text{seq}} S_2 \prec_{\rightarrow}^{\text{seq}} S_3$. We check that any direct order of $B(S_1, S_2, S_3)$ is indeed τ -balancing.

► **Lemma 10.** *A direct order $\prec_{\rightarrow}^{\text{seq}}$ of the bottleneck sequence $B(S_1, S_2, S_3)$ is τ -balancing.*

Proof. TOPROVE 3 ◀

3.2 Encoding NAE 3-Sat in Linear Degree Balancing

We now describe the reduction from NAE 3-SAT to LINEAR DEGREE BALANCING. We recall that a not-all-equal 3-clause is satisfied if it has at least one satisfied literal and at least one unsatisfied literal. The NAE 3-SAT remains NP-hard if each clause is on exactly three distinct *positive* literals, and every variable appears exactly four times positively (and zero times negatively) [5]. Let φ be any such n -variable NAE 3-SAT instance. As we will only deal with not-all-equal 3-clauses, we say that φ is *satisfiable* whenever it admits a truth assignment that, in each clause of φ , sets a (positive) literal to true and another (positive) literal to false. We will build an edge-weighted graph $H := H(\varphi)$ as follows.

Variables, clauses, and variable-clause incidence. For each variable x of φ , we add a vertex v_x (to $H(\varphi)$), the *vertex* of x . For each clause c of φ we add a vertex v_c , the *vertex* of c . For every clause c and every variable x in c , we add the edge $v_x v_c$ of weight λ . We add two sets of vertices $T = \{t_i : i \in [n]\}$ and $F = \{f_i : i \in [n]\}$, for *true* and *false*. For each t_i (resp. each f_i), we add a vertex \bar{t}_i (resp. a vertex \bar{f}_i), and the edge $t_i \bar{t}_i$ (resp. $f_i \bar{f}_i$) of weight $\tau - \lambda$. For each $i \in [n]$, let x_i be the i -th variable of φ . We add a vertex \bar{v}_{x_i} , and the edges $v_{x_i} t_i, v_{x_i} f_i, \bar{v}_{x_i} \bar{t}_i, \bar{v}_{x_i} \bar{f}_i$ each of weight λ .

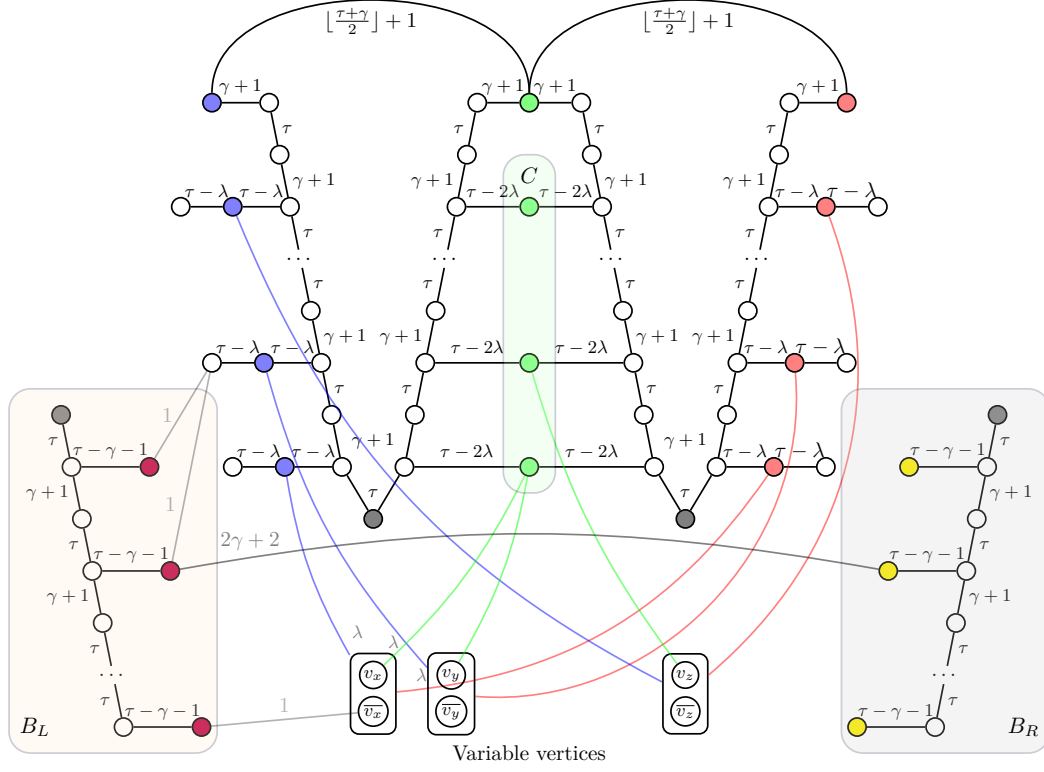
Bottleneck sequence $B(T, C, F)$. We then add a bottleneck sequence $B(T, C, F)$ where $C := \{v_c : c \text{ is a clause of } \varphi\}$, with weight $\tau - \lambda$ on every attachment incident to T or F , and weight $\tau - 2\lambda$ on every attachment incident to C . (This is allowed since $\gamma + 1 \leq \tau - 2\lambda \leq \tau - \lambda \leq \tau - \gamma - 1$.) We remind the reader that every attachment of the first terminals of the bottlenecks forming $B(T, C, F)$ has weight $\gamma + 1$. These three first terminals are extra vertices not in T , C , and F .

This could end the construction of H , but we want to impose an extra condition, which will later prove useful. Specifically, we want that all *but two* vertices have weight at least $\tau + \gamma + 1$ (both having weight τ). Let us call H' the edge-weighted graph built so far.

Weight padding. For each vertex $v \in V(H')$ of weight less than $\tau + \gamma + 1$, the *missing weight* of v is defined as $\tau + \gamma + 1$ minus the weight of v . Let p be the sum of missing weights of vertices of H' . Let X and Y be two sets each comprising p new vertices. We add a bottleneck B_L with terminals the vertices of X , and a bottleneck B_R with terminals the vertices of Y . Every attachment to B_L and B_R with an unspecified weight gets weight $\tau - \gamma - 1$. We add a perfect matching between X and Y with every edge of weight $2\gamma + 2$. Finally, for each vertex $v \in V(H')$, we link v by edges of weight 1 to t vertices of X , where t is the missing weight of v . We do so such that every vertex in X has exactly one neighbor in $V(H')$.

This completes the construction of H ; see Figure 4. We observe that H is triangle-free. This fact will significantly simplify some proof in Section 5 (although is not in any way crucial).

We check that the weight padding works as intended.



■ **Figure 4** Illustration of (H, ω) . Centered at the top is the bottleneck sequence $B(T, C, F)$. The vertices of X are in purple (left), and the vertices of Y are in yellow (right). The edges incident to the variable vertices that are drawn in blue, green, red all have weight λ . Not to overburden the figure, we have only drew *some* edges of the construction. Only one edge of the matching between X and Y is depicted, and the paddings of \bar{v}_x and of \bar{t}_2 are (partially) represented (weight-1 edges toward X). The clause corresponding to the bottommost vertex of C contains x, y and some other variable (not shown), while that of the second bottommost vertex of C contains z (and two other variables). The roots of bottlenecks are in gray. The leftmost and rightmost gray vertices are the only two vertices of weight less than $\tau + \gamma + 1$ (namely τ).

► **Lemma 11.** *Every vertex of H has weight at least $\tau + \gamma + 1$, except two vertices.*

Proof. TOPROVE 4 ◀

3.3 Preparatory lemmas

We will make use of the following two lemmas.

► **Lemma 12.** *For any $(\tau + \gamma)$ -balancing order \prec of $H(\varphi)$, for every clause $c = x \vee y \vee z$ of φ , $\{v_x, v_y, v_z\}$ surrounds v_c .*

Proof. TOPROVE 5 ◀

► **Lemma 13.** *For any $(\tau + \gamma)$ -balancing order \prec of $H(\varphi)$, for any variable x of φ , we either have $v_x \prec C$ or $C \prec v_x$.*

Proof. TOPROVE 6 ◀

3.4 Correctness of the reduction

We can now show that our reduction performs as announced.

► **Lemma 14.** *If $H(\varphi)$ admits a $(\tau + \gamma)$ -balancing order, then φ is satisfiable.*

Proof. TOPROVE 7 ◀

► **Lemma 15.** *If φ is satisfiable, then $H(\varphi)$ admits a τ -balancing order.*

Proof. TOPROVE 8 ◀

Theorem 3 is a direct consequence of Lemmas 14 and 15. The reason we “padded the degree” in $H(\varphi)$ will become apparent in the next section. We will observe that when φ is unsatisfiable, not only no linear order can “balance” the degrees, but no tree can either.

3.5 From linear orders to trees

As mim-width is defined via branch decompositions, we adapt the balancing order problem to trees. Consider an edge-weighted graph (H, ω) , and a tree T such that there exists a bijective map $f: V(H) \rightarrow V(T)$. Note that (T, f) is *not* a branch decomposition of H for two reasons: vertices of H are mapped to *all* nodes of T and not merely its leaves, and T is not necessarily a ternary tree (nor a rooted binary tree).

Each edge e of T defines a cut of H , which we denote (A_e, B_e) , where A_e is the preimage by f of one connected component of $T - e$, and B_e , of the other component. We say that (T, f) is a τ -balancing tree of (H, ω) if for any vertex $v \in V(H)$, for any edge $e \in E(T)$ incident to $f(v)$, the sum of the weights of edges (in $E(H)$) incident to v in the cut (A_e, B_e) is at most τ .

TREE DEGREE BALANCING

Parameter: τ

Input: An edge-weighted graph (H, ω) and a non-negative integer τ .

Question: Does (H, ω) admit a τ -balancing tree?

Note that any graph with a τ -balancing order also admits a τ -balancing tree, with T being the path of length $|V(H)|$, and f mapping the vertices of H along T in the τ -balancing order.

► **Theorem 16.** *Given an edge-weighted graph (H, ω) promised to satisfy either one of*
 ■ (H, ω) admits a τ -balancing order or
 ■ (H, ω) does not admit a $(\tau + \gamma)$ -balancing tree,
deciding which outcome holds is NP-hard.

Proof. TOPROVE 9 ◀

4 Degree Balancing to Linear Mim-Balancing/Tree Sim-Balancing

In this section, we show how to transfer the degree requirement of DEGREE BALANCING to the induced-matching requirement of MIM- and SIM-BALANCING.

4.1 The Mim-Balancing and Sim-Balancing problems

A *partitioned graph* is a pair (G, \mathcal{S}) where G is a graph and \mathcal{S} is a partition of $V(G)$. A *tree mapping* of a partitioned graph (G, \mathcal{S}) is a pair (T, f) where T is a tree and $f: \mathcal{S} \rightarrow V(T)$ is a bijection from the parts of \mathcal{S} to the vertices of T . When T is a path, we may call (T, f) a *path mapping* of \mathcal{S} .

We say that a cut (A, B) of G is an \mathcal{S} -cut if each set in \mathcal{S} is included in either A or B . Each edge $e \in E(T)$ in a tree mapping (T, f) of (G, \mathcal{S}) defines an \mathcal{S} -cut (A_e, B_e) of G : the union of the parts mapped to each component of $T - e$. The *sim-value* (resp. *mim-value*) of a tree mapping (T, f) of (G, \mathcal{S}) is the maximum taken over every edge $e \in E(T)$ of the maximum size of an induced (resp. semi-induced) matching between A_e and B_e . The *sim-balancing* (resp. *mim-balancing*) of (G, \mathcal{S}) is the minimum sim-value (resp. mim-value) among all possible tree mappings of (G, \mathcal{S}) . Similarly, the *linear sim-balancing* (resp. *mim-balancing*) is the minimum sim-value (resp. mim-value) among paths mappings.

TREE SIM-BALANCING (resp. TREE MIM-BALANCING)

Parameter: τ

Input: A partitioned graph (G, \mathcal{S}) and a non-negative integer τ .

Question: Does (G, \mathcal{S}) admit a tree mapping (T, f) of sim-value (resp. mim-value) τ ?

Note that even when \mathcal{S} is the finest partition $\{\{v\} : v \in V(G)\}$, this problem is not exactly MIM-WIDTH, as f also maps vertices to internal nodes of T , and T has no degree restriction. LINEAR MIM-BALANCING (or LINEAR SIM-BALANCING) is defined analogously except T is forced to be a path. We may use MIM-BALANCING to collectively refer to LINEAR MIM-BALANCING and TREE MIM-BALANCING; and similarly with SIM-BALANCING.

At the end of this section, we will have established the following.

► **Theorem 17.** *Let τ, γ be natural numbers satisfying Equation (1) and $\gamma > 50$. Given partitioned graphs (G, \mathcal{S}) such that:*

- *the linear mim-balancing of (G, \mathcal{S}) is at most $\tau + 50$, or*
- *the sim-balancing of (G, \mathcal{S}) is at least $\tau + \gamma$,*

deciding which of the two outcomes holds is NP-hard.

4.2 Encoding Degree Balancing in Mim/Sim-Balancing

Let (H, ω) be an instance of TREE DEGREE BALANCING with positive and integral weights. We build an instance of TREE MIM-BALANCING $G := G(H, \omega), \mathcal{S} := \mathcal{S}(H, \omega)$, as follows.

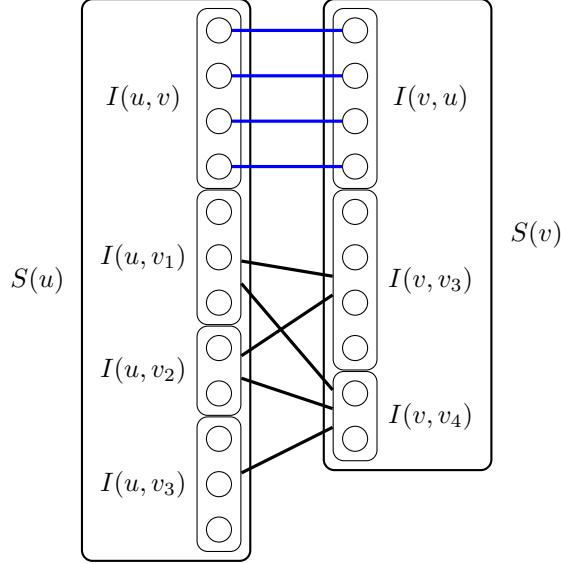
Construction of (G, \mathcal{S}) . For every vertex $u \in V(H)$ and every $v \in N_H(u)$, we add an independent set $I(u, v)$ of size $\omega(uv)$ to G . For each vertex $u \in V(H)$, we set

$$S(u) := \bigcup_{v \in N_H(u)} I(u, v).$$

Each $S(u)$ will remain an independent set in G . The partition \mathcal{S} is simply defined as $\{S(u) : u \in V(H)\}$.

We finish the construction by adding two kinds of edges in G , *matching edges* and *dummy edges*. For every pair of disjoint edges uv and xy of H , we add an edge between every vertex of $I(u, v)$ and every vertex of $I(x, y)$. All these edges are called *dummy*. For every $uv \in E(H)$, we add a maximum (perfect) induced matching between $I(u, v)$ and $I(v, u)$. All these edges are called *matching edges*. Observe that $\omega(uv) = \omega(vu)$ (H is undirected),

hence $|I(u, v)| = |I(v, u)|$ and the matching between $I(u, v)$ and $I(v, u)$ is indeed perfect. This concludes the construction of (G, \mathcal{S}) ; see Figure 5 for an illustration of the adjacencies between some $S(u)$ and $S(v)$.



■ **Figure 5** Adjacencies between $S(u)$ and $S(v)$. In this example, u has four neighbors v, v_1, v_2, v_3 , and v has three neighbors u, v_3, v_4 . The matching edges are in blue, the dummy edges are in black (edges between two boxes represent bicliques). Notice the non-edges between $I(u, v_3)$ and $I(v, v_3)$.

We notice that the configuration of the figure actually implies that uvv_3 is a triangle in H , which does not happen in graphs H produced by the previous reduction. However, we will not use that H is triangle-free in the current section, and Figure 5 shows the general behavior between $S(u)$ and $S(v)$. (For triangle-free graphs H , if $uv \in E(H)$, then there would instead be a biclique between $S(u) \setminus I(u, v)$ and $S(v) \setminus I(v, u)$, and if $uv \notin E(H)$, $I(u, v)$, $I(v, u)$, and the matching edges in between them would simply not exist.)

4.3 Preparatory lemmas

We will now prove some facts about the (semi-)induced matchings of the \mathcal{S} -cuts of G .

► **Lemma 18.** *Let (A, B) be an \mathcal{S} -cut of G . If there is no dummy edge between a vertex of $I(u, v) \subseteq A$ and one from $I(x, y) \subseteq B$, then $u = y$ or $v = x$ or $v = y$.*

Proof. TOPROVE 10 ◀

► **Lemma 19.** *Let (A, B) be an \mathcal{S} -cut of G . Assume there exists a semi-induced matching $M := \{e_1, \dots, e_m\}$ in G between A and B containing matching edges only. Then a single part of \mathcal{S} covers all the edges of M .*

Proof. TOPROVE 11 ◀

► **Lemma 20.** *Let (A, B) be an \mathcal{S} -cut of G . If $M := \{a_1b_1, \dots, a_tb_t\}$ is a semi-induced matching in G between A and B only made of dummy edges, and all the vertices a_i lie in the same part of \mathcal{S} included in A , then $t \leq 6$.*

Proof. TOPROVE 12 ◀

► **Lemma 21.** *Let (A, B) be a \mathcal{S} -cut of (G, \mathcal{S}) with a semi-induced matching $\{e_1, \dots, e_m\}$ between A and B . Then at least $m - 50$ edges among $\{e_1, \dots, e_m\}$ are matching edges.*

Proof. TOPROVE 13 ◀

4.4 Correctness of the reduction

We can now show the correctness of the reduction.

► **Lemma 22.** *If (G, \mathcal{S}) admits a tree mapping (T, f) of sim-value t , then (H, ω) admits a t -balancing tree.*

Proof. TOPROVE 14 ◀

► **Lemma 23.** *If H admits a τ -balancing order, then there is a path mapping (P, f) of mim-value at most $\tau + 50$.*

Moreover, for every cut (A_e, B_e) induced by an edge $e \in E(P)$ and every semi-induced matching M between A_e and B_e , M has at most τ matching edges and there exists $u \in V(H)$ such that $S(u)$ covers the matching edges of M .

Proof. TOPROVE 15 ◀

5 Mim/Sim-Balancing to Linear Mim-Width/Sim-Width

The next reduction uses two constants $a := 45$ and $b := 6\tau(\tau + \gamma) + 1$. With the announced values of $\tau = 1080$ and $\gamma = 135$, we have $b = 7873201$. We remark that the value of b will not affect the linear mim-width upper bound nor the sim-width lower bound. (The constant b should simply be that large to make our proofs work.)

Let (H, ω) be an instance of TREE DEGREE BALANCING where all edge weights are positive multiples of a and H is triangle-free. We build a graph G^* , such that if H has a τ -balancing order, then the linear mim-width of G^* is at most $\frac{a+1}{a}\tau + 107$; and if H has no $(\tau + \gamma)$ -balancing tree, then the sim-width of G^* is at least $\tau + \gamma$. We construct G^* from the instance $G := G(H), \mathcal{S} := \mathcal{S}(H)$ of TREE MIM-BALANCING from the previous reduction. Remember that $\mathcal{S} = \{S(u) : u \in V(H)\}$.

The main goal of this reduction is to obtain a graph G^* whose sim-width and linear mim-width are related to the sim-balancing and linear mim-balancing of (G, \mathcal{S}) , respectively. Observe that we cannot simply set $G^* := G$ since a layout (T, f) of G can scatter each $S(u)$ so that for each matching edge xy of G , x and y are placed at leaves of T sharing a neighbor in T . Consequently, the only cuts (A, B) induced by the edges of T with a matching edge between A and B are rather trivial (A or B is a singleton). Thus, the sim-width of G could be uncontrollably smaller than the sim-balancing of (G, \mathcal{S}) .

To prevent this, we design a gadget $\mathcal{G}(u)$ for each $u \in V(H)$ from b copies of $S(u)$. These gadgets ensure that any tree layout (T, f) of G^* of sim-value at most $\tau + \gamma - 1$ behaves similarly to a tree mapping of (G, \mathcal{S}) in the sense that for every $S(u)$, there is an edge e of T such that both sides of the induced cut (A_e, B_e) contain a copy of $S(u)$. Using this property, we prove that the sim-width of G^* is at least the sim-balancing of (G, \mathcal{S}) .

The final lemma from each of the two last subsections prove Theorem 2 (hence Theorem 1), which we restate here.

► **Theorem 24.** *Let τ, γ, a be natural numbers as previously defined. Given graphs G such that either*

- the linear mim-width of G is at most $\frac{a+1}{a}\tau + 107$, or
 - the sim-width of G is at least $\tau + \gamma + 1$,
- deciding which of the two outcomes holds is NP-hard.

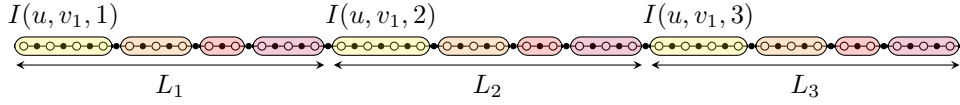
One can indeed check that with the announced values for τ, γ, a , we have $\frac{a+1}{a}\tau + 107 = 1211$ and $\tau + \gamma + 1 = 1216$.

5.1 Encoding Mim/Sim-Balancing in Mim/Sim-Width

We start with the description of a gadget for each vertex of H .

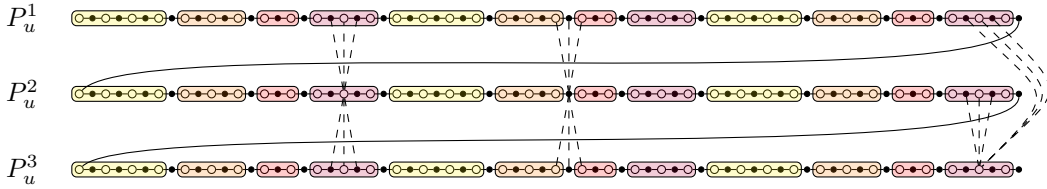
Construction of $\mathcal{G}(u)$. For each vertex $u \in V(H)$, the *gadget* of u , denoted by $\mathcal{G}(u)$, is a graph spanned by a path Q_u of length $2b \cdot |S(u)|$ made by concatenating b copies of a path P_u . The path P_u is built as follows. Recall that in the graph G , the set $S(u)$ partitions into $I(u, v_1) \uplus \dots \uplus I(u, v_k)$ where $\{v_1, \dots, v_k\} = N_H(u)$. Since all weights are multiples of a , $|I(u, v)|$ is a multiple of a for any edge $uv \in E(H)$. Hence we can write each $I(u, v)$ as a disjoint union $I(u, v, 1) \uplus \dots \uplus I(u, v, a)$ where each $I(u, v, i)$ has size $\frac{|I(u, v)|}{a}$.

We construct the path L_i whose vertex set is $I(u, v_1, i) \cup I(u, v_2, i) \cup \dots \cup I(u, v_k, i)$, and whose vertices occur in this order along L_i . We define L as the concatenation $L_1 L_2 \dots L_a$, i.e., the last vertex of L_i is made adjacent to the first vertex of L_{i+1} , for every $i \in [a-1]$. The path P_u is obtained from the 1-subdivision of L by adding a vertex adjacent to the last vertex of L_a ; see Figure 6.



■ **Figure 6** The path P_u for a vertex u with four neighbors v_1, v_2, v_3, v_4 , and $a = 3$. The sizes of $I(u, v_1)$, $I(u, v_2)$, $I(u, v_3)$, $I(u, v_4)$ are 12, 9, 6, 9, respectively; all divisible by a . The labels $I(u, v_1, \bullet)$ and L_\bullet refer to the white vertices, while P_u also comprises the subdivision vertices in black.

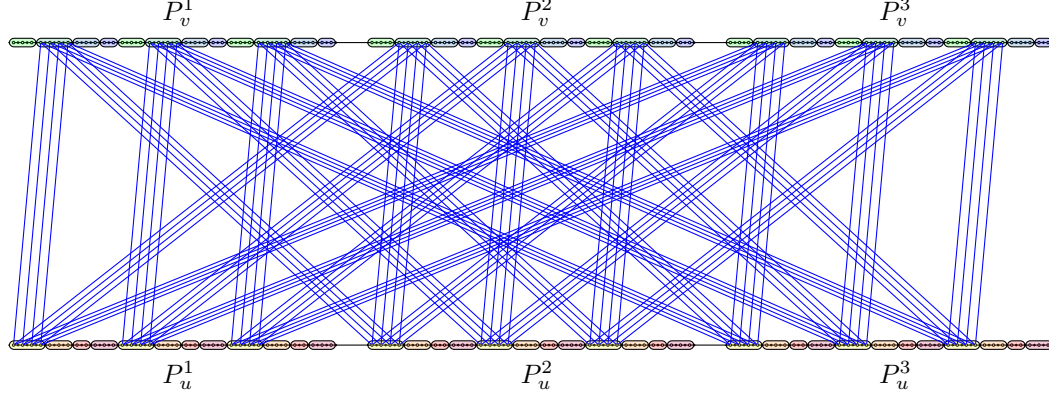
We obtain the path Q_u by concatenating b copies $P_u^1, P_u^2, \dots, P_u^b$ of P_u . Note that each vertex x of P_u has b copies x_1, \dots, x_b in Q_u ; for each $y \in \{x, x_1, \dots, x_b\}$, we denote by $\text{Copies}(y)$ the set $\{x_1, \dots, x_b\}$. The gadget $\mathcal{G}(u)$ is obtained from Q_u by adding an edge between every pair of vertices x, y in two distinct P_u^i, P_u^j except if y is in $N_{Q_u}[\text{Copies}(x)]$; see Figure 7.



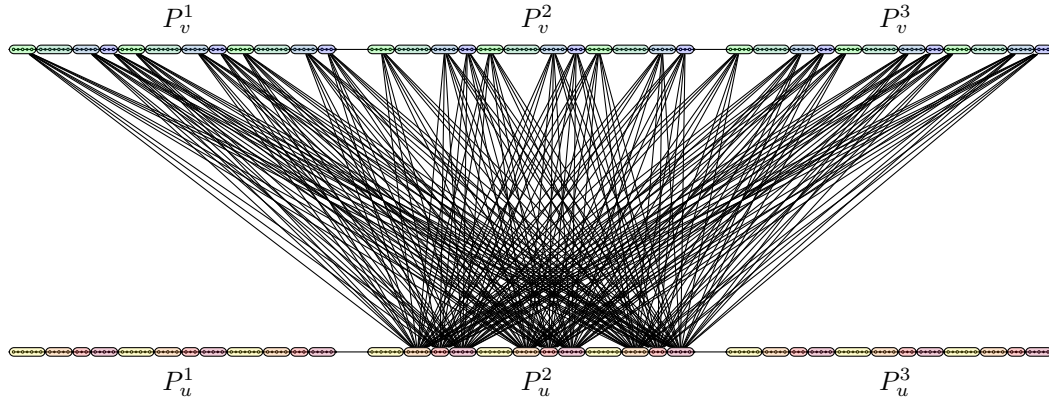
■ **Figure 7** The gadget $\mathcal{G}(u)$ for the path P_u of Figure 6 and $b = 3$. We drew the non-edges between distinct copies of P_u (dashed edges) incident to only three vertices (one subdivision vertex in P_u^2 , and two regular vertices in P_u^2 and P_u^3). Each path P_u^i remains induced in $\mathcal{G}(u)$.

Construction of G^* . Finally, we construct G^* as follows (based on the vertex set of H , and the edge set of G). For each vertex $u \in V(H)$, we add a gadget $\mathcal{G}(u)$ to G^* . For

every edge $xy \in E(G)$, we add the biclique between $\text{Copies}(x)$ and $\text{Copies}(y)$ in G^* . If xy is a matching edge of G , the added edges are also said *matching* (see Figure 8). Similarly if xy is a dummy edge, we call the added edges *dummy* (see Figure 9).



■ **Figure 8** The matching edges between $\mathcal{G}(u)$ and $\mathcal{G}(v)$ (with u and v two adjacent vertices in H).



■ **Figure 9** The dummy edges between $\mathcal{G}(u)$ and $\mathcal{G}(v)$ of Figure 8. An edge between two rounded boxes represents a biclique between the corresponding vertices of $I(\bullet, \bullet, \bullet)$ (which excludes the subdivision vertices). We only represented the bicliques incident to P_u^2 (P_u^1 and P_u^3 have the same adjacencies toward $\mathcal{G}(v)$).

5.2 Low linear mim-balancing of $(G, \mathcal{P}) \Rightarrow$ low linear-mim width of G^*

To upper bound the linear mim-width of G^* , we need the next lemma on $\mathcal{G}(u)$. For each vertex u of H , we define the *caterpillar layout* of $\mathcal{G}(u)$ as the *left-aligned* caterpillar (i.e, such that every right child is a leaf) layout with $|V(\mathcal{G}(u))|$ leaves bijectively labeled by $V(\mathcal{G}(u))$, in the order of Q_u from the first vertex of P_u^1 to the last vertex of P_u^b .

► **Lemma 25.** *Let u be a vertex of H , and (C, f) be the caterpillar layout of $\mathcal{G}(u)$. For every cut (A, B) of $\mathcal{G}(u)$ induced by an edge of C , the mim-value of (A, B) is at most 7.*

Proof. TOPROVE 16

◀

We can now conclude for this direction of the reduction.

► **Lemma 26.** *If (H, ω) admits a τ -balancing order, then the linear mim-width of G^* is at most $\frac{a+1}{a}\tau + 107$.*

Proof. TOPROVE 17 ◀

5.3 Low sim-width of $G^* \Rightarrow$ low sim-balancing of (G, \mathcal{P})

The main argument works as follows. We will prove that in any tree layout of G^* of small sim-value, one can associate to each gadget $\mathcal{G}(u)$ an edge e of the layout, such that a copy of P_u can be found on both sides of the cut defined by e . Since $\mathcal{G}(u)$ is “represented” by any of its copies of P_u , we will use e as where the vertices of $\mathcal{G}(u)$ should be moved to. With this in mind, we gradually build a tree mapping of G from a tree layout of G^* by “relocating” each gadget to the edge it is associated to.

We start with two technical lemmas.

► **Lemma 27.** *Let P_1, \dots, P_k be k paths with $k > t \cdot \max_{i \in [k]} |V(P_i)|$ such that for any $i \neq j \in [k]$ and any $\ell \in [\min(|V(P_i)|, |V(P_j)|) - 1]$, the ℓ -th edge of P_i is mutually induced with the ℓ -th edge of P_j . Then for any cut (A, B) of sim-value at most t , there exists at least one path P_i such that $V(P_i) \subseteq A$ or $V(P_i) \subseteq B$.*

Proof. TOPROVE 18 ◀

► **Lemma 28.** *Let P_1, \dots, P_k be k paths with $k > \lceil \frac{3t}{2} \rceil \cdot \max_{i \in [k]} |V(P_i)|$ such that for any $i \neq j \in [k]$ and any $\ell \in [\min(|V(P_i)|, |V(P_j)|) - 1]$, the ℓ -th edge of P_i is mutually induced with the ℓ -th edge of P_j . Then for any tripartition (A, B, C) of $V(P_1) \cup \dots \cup V(P_k)$ such that the cuts $(A, B \cup C)$, $(B, A \cup C)$ and $(C, A \cup B)$ have sim-value at most t , there exists at least one path P_i such that $V(P_i)$ is included in one set among A , B and C .*

Proof. TOPROVE 19 ◀

A *hybrid tree* of a partitioned graph (J, \mathcal{P}) is pair (T, f) where T is a tree and $f: V(J) \rightarrow V(T)$ is a map such that for any node $t \in T$, either $f^{-1}(t) \in \mathcal{P}$ or $|f^{-1}(t)| \leq 1$. As for tree mappings each edge $e \in E(T)$ in a hybrid tree of (J, \mathcal{P}) defines a cut (A_e, B_e) of J : the sets of vertices mapped to each component of $T - e$. The sim-value of the hybrid tree (T, f) is the maximum sim-value of all possible cuts (A_e, B_e) for $e \in E(T)$.

We recall that $\mathcal{S}^* = \{V(\mathcal{G}(u)) : u \in V(H)\}$. We observe that in the instances (H, ω) produced by the first reduction, every vertex has weight at most 2τ . Hence $\max_{u \in V(H)} |V(P_u)| \leq 4\tau$. We set $\alpha := \lceil \frac{b-1}{6\tau} \rceil - 1 = \tau + \gamma - 1$, where b is the constant introduced at the beginning of the section.

► **Lemma 29.** *Let (T, f) be a hybrid tree of (G^*, \mathcal{S}^*) of sim-value at most α and T subcubic. Then, for any vertex u of H , either*

- *there exists $t \in V(T)$ with $f^{-1}(t) = V(\mathcal{G}(u))$, or*
- *there exists an edge $e \in E(T)$ such that in the cut (A_e, B_e) induced by e in G^* , one can find a copy of P_u in both A_e and B_e .*

Proof. TOPROVE 20 ◀

We now define a *grouping* operation on triples (T, f, u) , where (T, f) is a subcubic hybrid tree of (G^*, \mathcal{S}^*) of sim-value smaller than $\frac{2(b-1)}{3 \max_{u \in V(H)} |V(P_u)|}$ (which is still very large compared to $\tau + \gamma$ by definition of b) and $u \in V(H)$, and denote it by $\text{group}(T, f, u)$. If there exists $t \in V(T)$ with $f^{-1}(t) = \mathcal{G}(u)$, then we set $\text{group}(T, f, u) := (T, f)$. Otherwise, Lemma 29 ensures that there is an edge $e \in E(T)$ such that a copy of P_u is in both sides of the cut defined by e . We then define $\text{group}(T, f, u) := (T', f')$, where

- T' is obtained from T by subdividing e , which adds a node, say, t_e , and
- f' satisfies that $f'(x) = f(x)$ whenever $x \notin V(\mathcal{G}(u))$, and $f'(x) = t_e$ otherwise.

Given an edge $e' \in E(T')$, the edge *corresponding* to e' in T is e' if e' is an edge of T , and e if e' is incident to t_e .

We make two observations on the grouping operation.

► **Observation 30.** *For any subcubic hybrid tree (T, f) of (G^*, \mathcal{S}^*) and any $u \in V(H)$, $\text{group}(T, f, u)$ is a subcubic hybrid tree of (G^*, \mathcal{S}^*) .*

► **Observation 31.** *Let (T, f) be a subcubic hybrid tree of (G^*, \mathcal{S}^*) and $u \in V(H)$. Let $(T', f') := \text{group}(T, f, u)$, $e \in E(T)$, and $e' \in E(T')$ corresponds to e in T . Then if (A_e, B_e) and $(A_{e'}, B_{e'})$ are the cuts of G^* defined by e and e' , respectively, we have*

- $A_{e'} \setminus V(\mathcal{G}(u)) = A_e \setminus V(\mathcal{G}(u))$,
- $B_{e'} \setminus V(\mathcal{G}(u)) = B_e \setminus V(\mathcal{G}(u))$,
- $V(\mathcal{G}(u)) \subseteq A_{e'}$ implies that A_e contains a copy of P_u , and
- $V(\mathcal{G}(u)) \subseteq B_{e'}$ implies that B_e contains a copy of P_u .

We can next show that a grouping can only decrease the sim-value.

► **Lemma 32.** *For any subcubic hybrid tree (T, f) of (G^*, \mathcal{S}^*) and any $u \in V(H)$, the sim-value of $\text{group}(T, f, u)$ is at most that of (T, f) .*

Proof. TOPROVE 21 ◀

We are now equipped to turn hybrid trees G^* into tree mappings of G^* no greater sim-value.

► **Lemma 33.** *If (G^*, \mathcal{S}^*) admits tree layout (T, f) of sim-value at most α , then (G^*, \mathcal{S}^*) admits a tree mapping (T', f') of sim-value at most the sim-value of (T, f) .*

Proof. TOPROVE 22 ◀

We can conclude.

► **Lemma 34.** *Let (T, f) be a tree layout witnessing that the sim-width of G^* is at most $\tau + \gamma$. Then the tree sim-balancing of (G, \mathcal{S}) is at most $\tau + \gamma$.*

Proof. TOPROVE 23 ◀

References

- 1 Rémy Belmonte and Martin Vatshelle. Graph classes with structured neighborhoods and algorithmic applications. *Theor. Comput. Sci.*, 511:54–65, 2013. URL: <https://doi.org/10.1016/j.tcs.2013.01.011>, doi:10.1016/J.TCS.2013.01.011.
- 2 Benjamin Bergougnoux, Jan Dreier, and Lars Jaffke. A logic-based algorithmic meta-theorem for mim-width. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 3282–3304. SIAM, 2023. URL: <https://doi.org/10.1137/1.9781611977554.ch125>, doi:10.1137/1.9781611977554.CH125.
- 3 Benjamin Bergougnoux, Tuukka Korhonen, and Igor Razgon. New width parameters for independent set: One-sided-mim-width and neighbor-depth. In Daniël Paulusma and Bernard Ries, editors, *Graph-Theoretic Concepts in Computer Science - 49th International Workshop, WG 2023, Fribourg, Switzerland, June 28-30, 2023, Revised Selected Papers*, volume 14093 of *Lecture Notes in Computer Science*, pages 72–85. Springer, 2023. doi:10.1007/978-3-031-43380-1_6.

- 4 Benjamin Bergougnoux, Charis Papadopoulos, and Jan Arne Telle. Node multiway cut and subset feedback vertex set on graphs of bounded mim-width. *Algorithmica*, 84(5):1385–1417, 2022. URL: <https://doi.org/10.1007/s00453-022-00936-w>, doi:10.1007/S00453-022-00936-w.
- 5 Andreas Darmann and Janosch Döcker. On a simple hard variant of Not-All-Equal 3-Sat. *Theor. Comput. Sci.*, 815:147–152, 2020. URL: <https://doi.org/10.1016/j.tcs.2020.02.010>, doi:10.1016/J.TCS.2020.02.010.
- 6 Eduard Eiben, Robert Ganian, Thekla Hamm, Lars Jaffke, and O-joung Kwon. A unifying framework for characterizing and computing width measures. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 63:1–63:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. URL: <https://doi.org/10.4230/LIPIcs.ITCS.2022.63>, doi:10.4230/LIPIcs.ITCS.2022.63.
- 7 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.
- 8 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.
- 9 Lars Jaffke, O-joung Kwon, Torstein J. F. Strømme, and Jan Arne Telle. Mim-width III. graph powers and generalized distance domination problems. *Theor. Comput. Sci.*, 796:216–236, 2019. URL: <https://doi.org/10.1016/j.tcs.2019.09.012>, doi:10.1016/J.TCS.2019.09.012.
- 10 Lars Jaffke, O-joung Kwon, and Jan Arne Telle. Mim-Width I. Induced path problems. *Discret. Appl. Math.*, 278:153–168, 2020. URL: <https://doi.org/10.1016/j.dam.2019.06.026>, doi:10.1016/J.DAM.2019.06.026.
- 11 Yota Otachi, Akira Suzuki, and Yuma Tamura. Finding induced subgraphs from graphs with small mim-width. In Hans L. Bodlaender, editor, *19th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2024, June 12-14, 2024, Helsinki, Finland*, volume 294 of *LIPIcs*, pages 38:1–38:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. URL: <https://doi.org/10.4230/LIPIcs.SWAT.2024.38>, doi:10.4230/LIPIcs.SWAT.2024.38.
- 12 Sigve Hortemo Sæther and Martin Vatshelle. Hardness of computing width parameters based on branch decompositions over the vertex set. *Electron. Notes Discret. Math.*, 49:301–308, 2015. URL: <https://doi.org/10.1016/j.endm.2015.06.041>, doi:10.1016/J.ENDM.2015.06.041.
- 13 Sigve Hortemo Sæther and Martin Vatshelle. Hardness of computing width parameters based on branch decompositions over the vertex set. *Theor. Comput. Sci.*, 615:120–125, 2016. URL: <https://doi.org/10.1016/j.tcs.2015.11.039>, doi:10.1016/J.TCS.2015.11.039.
- 14 Martin Vatshelle. New width parameters of graphs. 2012.