Twin-width : lower bounds and approximation algorithms

Hugues Déprés

THESE

en vue de l'obtention du grade de Docteur, délivré par

l'ECOLE NORMALE SUPERIEURE DE LYON

Ecole Doctorale N° 512
École Doctorale en Informatique et Mathématiques de Lyon

Discipline : Informatique

Soutenue publiquement le 27/06/2024, par :

Hugues DÉPRÉS

Twin-width : lower bounds and approximation algorithms

Twin-width : bornes inférieures et algorithmes d'approximations

---

Devant le jury composé de :

GAVOILLE, Cyril	Professeur	LABRI, Université de Bordeaux	Rapporteur
THILIKOS, Dimitrios	DR CNRS	LIRMM	Rapporteur
PARREAU, Aline	CR CNRS	LIRIS	Examinatrice
LIEDLOFF, Mathieu	Professeur	LIFO, Université d'Orléans	Examinateur
BONNET, Édouard	CR-HDR CNRS	LIP, ENS de Lyon	Directeur de thèse

Remerciements

Je souhaiterais d'abord remercier Édouard : merci de m'avoir proposé ce sujet sur la twin-width et merci pour ton accompagnement. C'était très intéressant de pouvoir travailler sur un sujet suscitant l'engouement. Merci aussi à toi, à Stéphan et à Rémi d'avoir consacré autant de temps à partager vos connaissances sur la twin-width et sur la théorie des graphes. Merci Nicolas d'avoir accepté de t'occuper de la partie administrative de ma thèse malgré tes nombreuses autres responsabilités. Je remercie tous les autres membres de l'équipe MC2 avec qui j'ai pu discuter de graphes et d'autres choses au quotidien : Colin, Julien, Pegah, Pierre, Carl et Dibyayan.

Je voudrais remercier Cyril Gavoille et Dimitrios Thilikos pour avoir accepté d'être rapporteur de cette thèse. Vos remarques et vos conseils m'ont permis d'améliorer le manuscrit. Merci aussi à Aline Parreau et Mathieu Liedloff d'avoir accepté de faire partie de mon jury de thèse.

Merci à tous les acteurs de la communauté de la théorie des graphes qui m'ont invité à travailler avec eux pendant la thèse : Théo Pierron, Marthe Bonamy, Louis Esperet, Eun Jung Kim, Pierre Aboulker, Claire Hilaire, Alexandra Wesolek, Nicolas Bousquet, Laurent Feuilloley, Mamadou Moustapha Kanté, Sergei Norin, Paul Seymour, David Wood, Maël Dumas, Hoang La, Christophe Paul, Pierre Charbit, Reza Naserasr. Je remercie aussi le personnel administratif du LIP qui a rendu certaines de ces rencontres possibles. Merci à Russ Harmer pour son soutien administratif. Merci à Guillaume Hanrot pour ses conseils sur mon projet professionnel.

Merci aux amateurs de la pause café, aux amis des différents clubs de l'ENS et aux anciens camarades de promo grâce à qui (à cause de qui?) je n'ai pas fait que de la recherche à Lyon.

Enfin je souhaiterais remercier ma famille et en particulier mes parents qui m'ont aidé et soutenu durant ces 3 années.

Résumé

Cette thèse porte sur l'étude de nouvelles décomposition de graphes, notamment la twin-width qui est un nouveau paramètre permettant de mesurer la complexité d'un graphe, défini en 2020 par Eun Jung Kim, Stéphan Thomassé, Rémi Watrigant et Édouard Bonnet. C'est un paramètre qui s'est révélé être important dans la compréhension de la structure des graphes et de leurs algorithmes. Dans cette thèse, on montre que, décider si la twin-width d'un graphe est au plus 4, est un problème NP-complet. On réussit en effet à contrôler suffisamment la séquence de contraction pour simuler un circuit booléen. Cette réduction comporte notamment une construction permettant de faire apparaître certain trigraphes particuliers dans la séquence de contraction d'un graphe. On obtient ainsi des bornes inférieures sur la difficulté de calculer la twin-width. Dans cette thèse, on cherche aussi à borner la twin-width de plusieurs classes de graphes. On montre ainsi que la twin-width des longues subdivisions d'un graphe est au plus 4. On montre aussi l'optimalité de la borne supérieure sur la twin-width à treewidth borné. Cette construction se généralise aussi à d'autres paramètres comme la twin-width orienté. Enfin on étudie les algorithmes d'approximation à twin-width borné. Pour le problème du Stable Maximum à twin-width borné, on propose un algorithme de $O(1)$-approximation en temps $O(2^{\sqrt{n}})$. On construit ensuite un schéma qui permet d'améliorer la complexité de l'algorithme au prix de diminuer le facteur d'approximation. On obtient une n^{ε}-approximation en temps polynomial. Cette méthode s'applique aussi au problème du Coloriage Minimum et à celui du Couplage Induit Maximum.

Abstract

This thesis focuses on the study of new graph decompositions, in particular twin-width, which is a new parameter for measuring the complexity of a graph, defined in 2020 by Eun Jung Kim, Stéphan Thomassé, Rémi Watrigant and Édouard Bonnet. It turns out to be an important parameter for understanding the structure of graphs and their algorithms. In this thesis, we show that deciding whether the twin-width of a graph is at most 4 is an NP-complete problem. Indeed, we succeed in controlling the contraction sequence sufficiently to simulate a Boolean circuit. In particular, this reduction involves a construction that forces certain particular trigraphs to appear in the contraction sequence of a graph. This gives lower bounds on the difficulty of computing twin-width. In this thesis, we also seek to bound the twin-width of several classes of graphs. We show that the twin-width of the long subdivisions of a graph is at most 4. We also show the optimality of the upper bound on twin-width when treewidth is bounded. This construction is also generalized to other parameters such as oriented twin-width. Finally, we study approximation algorithms for bounded twin-width. For the Maximum Independent Set problem on graphs of bounded twin-width, we propose an $O(1)$-approximation algorithm in $O(2^{\sqrt{n}})$ time. We then construct a scheme that improves the complexity of the algorithm at the cost of reducing the approximation factor. The result is an n^{ε}-approximation algorithm in polynomial-time. This method can also be applied to the Minimum Vertex Coloring problem and the Maximum Induced Matching problem.

Introduction

How to determine if a problem is easy or hard to solve? Is an object simple or complex? These are fundamental questions in computer science and mathematics. To tackle these questions, one can look for properties that guarantee the simplicity of an object. But if we want a more refined, more gradual approach, we need a parameter: a quantity that varies depending on the object and that characterizes a dimension of the problem. We will then be able to measure the complexity according to this parameter. For graphs, natural parameters are the number of vertices and the number of edges, since they represent the complexity of the standard representation of graphs. But these parameters do not describe how the vertices are connected, meaning what patterns and properties the layout of the edges has.

There are many graph parameters describing more precisely these connections. A simple useful one is the maximum degree. Indeed, take the problem Clique where we are given a graph G and an integer k, and we want to know if there is a subset of k vertices such that every pair of these vertices is connected by an edge. This problem is known to be NP-hard, and in particular there is no known polynomial-time algorithm for solving this problem. However, if we restrict ourselves to graphs whose maximum degree is bounded by a constant d, then the problem is much easier to solve. For each vertex, we can try to make a clique with each subset of its neighbors. There are at most 2^d such subsets and we only need to check d^2 edges to know if a given subset forms a clique. Since we do this once for each vertex, there is a linear algorithm solving Clique when d is bounded.

There has been a fundamental work to improve existing parameters and find new ones that better describe the structural properties, better measure the algorithmic difficulty of the graph, and remains easy to compute. One of the most recent step in this line of work is twin-width, the parameter we study in this thesis.

Before presenting twin-width, let us determine what kind of properties we expect from such a parameter. For this purpose let us introduce a classical parameter: treewidth. The treewidth of a graph is a parameter that measures how similar to a tree a graph is. There are many results about the structure of graphs that involve treewidth, but we will only focus on two major algorithmic results. The first one is Courcelle's theorem [1] which states that on graphs of bounded treewidth, given a MSO_2-logic formula, meaning a formula where one can quantify over vertex sets and edge sets, there is a linear time algorithm deciding it. For example, the Hamiltonian cycle problem or the 3-colorability can be expressed as an MSO_2-formula. Indeed, for 3-colorability, we want to know if there exists 3 sets of vertices such that every vertex is in exactly one of the sets and such that no two vertices of the same set are adjacent. Furthermore, the converse of Courcelle's theorem also holds, meaning that treewidth is the most general parameter that has this property.

A point we left out is that for such algorithms to work, you need to input to these

algorithms a tree-decomposition, that is, the object that witnesses an upper bounds of the treewidth. This tree decomposition is like the schematic or a formula that explains why the graph is simple. The key now is to find it, and this is where the second result comes in handy. On graphs of bounded treewidth, Bodlaender's algorithm output such a decomposition in linear time [2]. Combining the two, on graphs of bounded treewidth, there are linear algorithms to solve many hard problems. Although the result of this theorem is very strong, requiring bounded treewidth is rather restrictive, for example it excludes grids. So the question now is, can we get similar results for more general parameters? And which parameter corresponds to first-order formulas? Twin-width is an important step towards answering these questions.

The idea for this parameter was born in 2014 in a paper from Guillemot and Marx [3]. This work focuses on permutations and in particular on deciding whether a permutation contains a specific pattern. The authors construct a new width parameter for permutations, to which is associated a decomposition that allows them to solve their problem. They had the intuition that it might be interesting to apply this parameter to graphs. In fact the decomposition is, to quote them : "more properly described as a construction scheme that maintains a notion of bounded-degreeness through-out the process."

A few years later, Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé and Rémi Watrigant adapted the parameter for graphs [4]. This immediately worked very well, generalizing several known classes of graphs. And the "decomposition", which we will, from now on, call the contraction sequence allows, given a first-order formula, to find a linear time algorithm that takes a graph of bounded twin-width and tells if it satisfies the formula. There are fewer problems that can be expressed as FO-formula than MSO_2-formula, but there are many more graphs of bounded twin-width than of bounded treewidth. Contrary to Courcelle's theorem, the converse is not true, meaning there are classes of graphs of unbounded twin-width, for which such an algorithm exists. For example the class of graphs of maximum degree 3 [5]. Note that, for this class, there is no known construction of graphs of arbitrary twin-width, only a counting argument showes their existence.

From there, the study of twin-width branches out in several ways. First there have been several width-measures constructed from twin-width [6, 7]. One of them is the adaptation of twin-width to ordered graphs, graphs where the vertices are given with an order. It happens that in this case we have the same situation as in the case of treewidth: the equivalence between having a fast FO-model checking algorithm and having bounded ordered twin-width, and moreover an efficient algorithm to compute a contraction sequence on graphs of bounded ordered twin-width[8].

Another topic is the computation of twin-width. For some classes of graphs, we know the exact value of twin-width, for other classes we only know bounds. There are many works trying to improve these bounds. For example there are a series of papers trying to bound the twin-width of planar graphs [9, 10, 11, 6, 12], which is currently known to be between 7 and 8. In this thesis, we will contribute by showing the tightness of some upper bounds on twin-width, in particular the one on graphs of bounded treewidth. There is also the question of computing twin-width on general graphs. There is no known algorithm for computing twin-width that is better than the exhaustive search. In Chapter 2 we will show that the problem is NP-hard ruling out a polynomial-time algorithm to compute twin-width. The problem is still NP-hard on graphs of bounded twin-width.

A last field of study is efficient algorithms on graphs of bounded twin-width. Apart from problems that can be expressed as a first-order formula, there may be other problems

that twin-width can help to solve. In particular, we can hope for better approximation algorithms. The only known result before this work is a polynomial approximation algorithm for Minimum Dominating Set. In the following, we will continue the study of approximation algorithms parameterized by twin-width. More precisely, we will show that for Maximum Independent Set and Coloring, on graphs of bounded twin-width, there exist approximation algorithms that run faster than what is possible in the general case.

Organization of the thesis

This thesis is divided into three parts. The first part introduces the necessary definitions for twin-width. It also contains a construction showing that long subdivisions of any graph have twin-width at most 4. Finally we will give classes of graphs whose twin-width is exponential in the following parameters : treewidth, oriented twin-width and grid number.

The second part will be devoted to prove the NP-hardness of computing twin-width. We will do a reduction from 3-SAT and introduce several gadgets. We will also show how we can construct a graph whose contraction sequence will be forced to contain a given trigraph.

The last part will study several problems on graphs parameterized by twin width. While these are known to be hard to approximate in general, we will show that on these graphs, some of them have faster approximation algorithms while others remain hard.

Publications

This work led to the following publications :

- Pierre Bergé, Édouard Bonnet, and Hugues Déprés. Deciding Twin-Width at Most 4 Is NP-Complete. In 49th International Colloquium on Automata, Languages, and Programming (ICALP 2022). Leibniz International Proceedings in Informatics (LIPIcs), Volume 229, pp. 18:1-18:20, Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2022)
- Édouard Bonnet, Hugues Déprés, Twin-width can be exponential in treewidth, Journal of Combinatorial Theory, Series B, Volume 161, 2023, Pages 1-14
- Pierre Bergé, Édouard Bonnet, Hugues Déprés, and Rémi Watrigant. Approximating Highly Inapproximable Problems on Graphs of Bounded Twin-Width. In 40th International Symposium on Theoretical Aspects of Computer Science (STACS 2023). Leibniz International Proceedings in Informatics (LIPIcs), Volume 254, pp. 10:1-10:15, Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2023)

Unrelated to this work, but still during my thesis, I've also contributed to the following article :

- Marthe Bonamy, Édouard Bonnet, Hugues Déprés, Louis Esperet, Colin Geniet, Claire Hilaire, Stéphan Thomassé, Alexandra Wesolek, Sparse graphs with bounded induced cycle packing number have logarithmic treewidth, Journal of Combinatorial Theory, Series B, Volume 167, 2024, Pages 215-249,

Contents

Chapter 1

The twin-width of graphs

Contents

In this chapter we present the necessary definitions for twin-width and some classical properties. We will also present two constructions that improve the bounds on twin-width for several classes of graphs.

1.1 Definitions and observations

A *trigraph* is a graph with some edges colored black, and some colored red. A (vertex) *contraction* consists of merging two (non-necessarily adjacent) vertices, say, u, v into a vertex w, and keeping every edge wz black if and only if uz and vz were previously black edges. The other edges incident to w become red (if not already), and the rest of the trigraph stays the same. A *contraction sequence* of an n-vertex graph G is a sequence of trigraphs $G = G_n, \dots, G_1 = K_1$ such that G_i is obtained from G_{i+1} by performing one contraction. A *d-sequence* is a contraction sequence where all the trigraphs have red degree at most d. The *twin-width* of G, denoted by $\mathrm{tww}(G)$, is then the minimum integer d such that G admits a d-sequence. See fig. 1.1 for an example of a graph admitting a 2-sequence.

In the following we will first define partition sequences, which is an alternative approach to contraction sequences. Then we will detail the definition using contraction sequences.

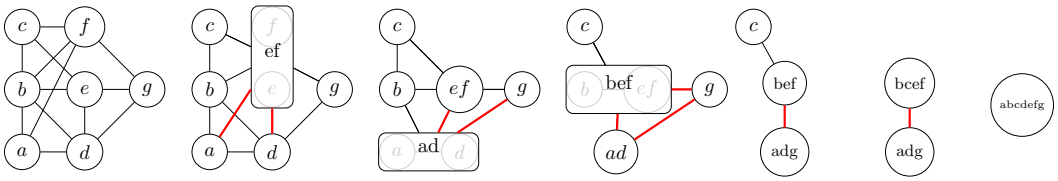


Figure 1.1: A 2-sequence witnessing that the initial graph has twin-width at most 2.

Finally, we will look at how common operations interact with twin-width.

The reason we give two (equivalent) definitions of twin-width is that both viewpoints are incomparably useful and convenient. The definition using contraction sequences emphasizes the fact that twin-width is obtained by a sequence of local changes of the graphs. To express each transformation we only need to look at the current neighborhood of the two contracted vertices. While this can simplify the manipulation of twin-width by limiting the number of factors to consider, it "loses" some parts of the information contained in the original graph; namely the exact adjacencies between non-homogeneous parts, and the names of the original vertices. Thus, we will define partitions sequences that make it easier to infer and prove the constraints that the original graph imposes on the sequence.

Note that twin-width can also be extended to matrices over a finite alphabet (in an unordered [4], or an ordered setting [8]), and hence to any binary structure.

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 $[1, i]$. We use the standard graph-theoretic notations: $V(G)$ denotes the vertex set of a graph G, $E(G)$ denotes its edge set, $G[S]$ denotes the subgraph of G induced by S, etc.

1.1.1 Partitions sequences

The *twin-width* of a graph, introduced in [4], can be defined in the following way (complementary to the one given above). A *partition sequence* of an n-vertex graph G, is a sequence $\mathcal{P}_n, \dots, \mathcal{P}_1$ of partitions of its vertex set $V(G)$, such that $\mathcal{P}_n$ is the set of singletons $\{\{v\} \ : \ v \in V(G)\}$, $\mathcal{P}_1$ is the singleton set $\{V(G)\}$, and for every $2 \leqslant i \leqslant n$, $\mathcal{P}_{i-1}$ is obtained from $\mathcal{P}_i$ by merging two of its parts into one. Two parts P, P' of a same partition $\mathcal{P}$ of $V(G)$ are said *homogeneous* if either every pair of vertices $u \in P, v \in P'$ are non-adjacent, or every pair of vertices $u \in P, v \in P'$ are adjacent. Two non-homogeneous parts are also said *red-adjacent*. The *red degree* of a part $P \in \mathcal{P}$ is the number of other parts of $\mathcal{P}$ which are red-adjacent to P. Finally the twin-width of G, denoted by $\mathrm{tww}(G)$, is the least integer d such that there is partition sequence $\mathcal{P}_n, \dots, \mathcal{P}_1$ of G with each part P of each $\mathcal{P}_i$ $(1 \leqslant i \leqslant n)$ being homogeneous to every part of $\mathcal{P}_i \setminus \{P\}$ but at most d.

Given a graph G and a partition $\mathcal{P}$ of $V(G)$, the *quotient graph* of G with respect to $\mathcal{P}$ is the graph with vertex set $\mathcal{P}$, where PP' is an edge if there is $u \in P$ and $v \in P'$ such that $uv \in E(G)$. Given a (tri)graph G and a partition $\mathcal{P}$ of $V(G)$, the *quotient trigraph* $G/\mathcal{P}$ is the trigraph with vertex set $\mathcal{P}$, where PP' is a black edge if these two parts are fully adjacent – for every $u \in P$ and every $v \in P'$, $uv \in E(G)$ –, and a red edge if either there is $u \in P$ and $v \in P'$ such that $uv \in R(G)$, or there is $u_1, u_2 \in P$ and $v_1, v_2 \in P'$ such that $u_1v_1 \in E(G)$ and $u_2v_2 \notin E(G)$.

1.1.2 Contractions sequences

We will start by making the definitions presented at the beginning of the chapter more formal. For a *trigraph* G we denote its vertex set by $V(G)$, its black edge set by $E(G)$ and its red edge set by $R(G)$. A *contraction* in a trigraph G replaces a pair of (non-necessarily adjacent) vertices $u, v \in V(G)$ by one vertex w that is linked to $G - \{u, v\}$ in the following way to form a new trigraph G'. For every $z \in V(G) \setminus \{u, v\}$, $wz \in E(G')$ whenever $uz, vz \in E(G)$, $wz \notin E(G') \cup R(G')$ whenever $uz, vz \notin E(G) \cup R(G)$, and $wz \in R(G')$, otherwise. Its *red graph* $(V(G), R(G))$ may be denoted $\mathcal{R}(G)$, and *total graph* $(V(G), E(G) \cup R(G))$, $\mathcal{T}(G)$. The *red degree* of a trigraph is the degree of its red graph. We say that $u \in V(G)$ is a *black neighbor* (respectively *red neighbor*) of $v \in V(G)$

when $(u,v) \in E(G)$ (respectively $(u,v) \in R(G)$). A trigraph G' is an *induced subtrigraph* of trigraph G if $V(G') \subseteq V(G)$, $E(G') = E(G) \cap \binom{V(G')}{2}$, and $R(G') = R(G) \cap \binom{V(G')}{2}$. Then we say that G is a supertrigraph of G', and we may also denote G' by $G[V(G')]$. A partial contraction sequence of an n-vertex (tri)graph G (to a trigraph H) is a sequence of trigraphs $G = G_n, \cdots, G_t = H$ for some $t \in [n]$ such that G_i is obtained from G_{i+1} by performing one contraction. A (complete) contraction sequence is such that $t = 1$, that is, H is the 1-vertex trigraph. A *d-sequence* $\mathcal{S}$ of G is a contraction sequence of G in which the red graph of every trigraph of $\mathcal{S}$ has maximum degree at most d.

We naturally consider the trigraph G_j to come *after* (resp. *before*) $G_{j'}$ if $j < j'$ (resp. $j > j'$). Thus when we write *the first trigraph of the sequence $\mathcal{S}$ to satisfy X* (or *the first time a trigraph of $\mathcal{S}$ satisfies X*) we mean the trigraph G_j with largest index j among those satisfying X. The same goes for partition sequences. A *(partial) d-sequence* is a (partial) contraction sequence where all the trigraphs have red degree at most d. In the above definitions of twin-width, nothing prevents the starting structure G to be a trigraph itself. We may then talk about the twin-width of a trigraph.

The definition using partitions sequences is equivalent to the one given above by considering the quotient trigraphs. Indeed, going from $G/\mathcal{P}_{i+1}$ to the one of $G/\mathcal{P}_i$ corresponds to the contraction operation described above. To navigate between these two worlds, and keep the proofs compact, we use the following notations and vocabulary. We assume that there is a partial contraction sequence from (tri)graph G to trigraph H. If u is a vertex of H, then $u(G)$ denotes the set of vertices eventually contracted into u in H. We denote by $\mathcal{P}(H)$ the partition $\{u(G) : u \in V(H)\}$ of $V(G)$. If G is clear from the context, we may refer to a *part* of H as any set in $\{u(G) : u \in V(H)\}$. We may say that two parts $y(G), z(G)$ of $\mathcal{P}(H)$ are *in conflict* if $yz \in R(H)$. We say that a contraction of two vertices $u, u' \in V(H)$ *involves* a vertex $v \in V(G)$ if $v \in u(G)$ or $v \in u'(G)$. A contraction *involves* a pair of vertices v, v' if $v \in u(G)$ and $v' \in u'(G)$ (or $v \in u'(G)$ and $v' \in u(G)$). Observe that the two vertices should appear in two distinct parts. By extension, we may say that a contraction *involves* a set S, if it involves a vertex of S, or a pair of sets S, T if it involves a pair in $S \times T$.

1.1.3 Graph operations and twin-width sequences

Twin-width can only decrease when taking induced subtrigraph.

Observation 1. *Let G' be an induced subtrigraph of trigraph G. Then $tww(G') \leqslant tww(G)$.*

A trigraph H is a *cleanup* of another trigraph G if $V(H) = V(G)$, $R(H) \subseteq R(G)$, and $E(G) \subseteq E(H) \subseteq E(G) \cup R(G)$. That is, H is obtained from G by turning some of its red edges into black edges or non-edges. We further say that H is *full cleanup* of G if H has no red edge, and thus, is considered as a graph. Note that the total graph $\mathcal{T}(G)$ and the *black graph* $(V(G), E(G))$ of a trigraph G are extreme examples of full cleanups of G.

It can be observed that removing red edges can only decrease the twin-width, since the contraction sequence for the initial trigraph works at least as well for the resulting trigraph.

Observation 2. *Any d-sequence of any trigraph is a d-sequence for any of its cleanups.*

Trees admit a simple 2-sequence, that gives a d-sequence on red trees of degree at most d.

Lemma 3 ([4])**.** *Every (black) tree has twin-width at most 2. Every red tree has twin-width at most its maximum degree.*

Proof. Root the tree arbitrarily. Contract two leaves with the same parent whenever possible. If not possible, contract a deepest leaf with its parent. Observe that this cannot result in red degree larger than the maximum of 2 and the initial degree of the tree. □

1.2 Bounds on twin-width

Graph classes with bounded twin-width include graphs with bounded treewidth, bounded clique-width, K_t-minor free graphs, strict subclasses of permutation graphs, map graphs, bounded-degree string graphs [4], segment graphs with no $K_{t,t}$ subgraph, visibility graphs of 1.5D terrains without large half-graphs, visibility graphs of simple polygons without large independent sets [13], as well as $\Omega(\log n)$-subdivisions of n-vertex graphs, classes with bounded queue number or bounded stack number, and some classes of cubic expanders [5].

Despite their apparent generality, classes of bounded twin-width are small [5], χ-bounded [14], even quasi-polynomially χ-bounded [15], preserved (albeit with a higher upper bound) by first-order transductions [4], and by the usual graph products when one graph has bounded degree [16, 5], have VC density 1 [17, 18], admit, when $O(1)$-sequences are given, a fixed-parameter tractable first-order model checking [4], an (almost) single-exponential parameterized algorithms for various problems that are W[1]-hard in general [14], as well as a parameterized fully-polynomial linear algorithm for counting triangles [19], an (almost) linear representation [20], a stronger regularity lemma [18], etc.

In all these applications, the upper bound on twin-width, although somewhat hidden in the previous paragraph, plays a role. There is then an incentive to obtain as low as possible upper bounds on particular classes of bounded twin-width. To give one concrete algorithmic example, an independent set of size k can be found in time $O(k^2d^{2k}n)$ in an n-vertex graph given with a d-sequence [14]. This is relatively practical for moderate values of k, with the guarantee that d is below 10, but not when d is merely upperbounded by 10^{10}. Another motivating example: triangle-free graphs of twin-width at most d are $d+2$-colorable [14], a stronger fact in the former case than in the latter.

In that line of work, Balabán and Hlinený show that posets of width k (i.e., with antichains of size at most k) have twin-width at most $9k$ [21]. Unit interval graphs have twin-width at most 2 [14], and proper k-mixed-thin graphs (a recently proposed generalization of unit interval graphs) have twin-width $O(k)$ [22]. Schidler and Szeider report the (exact) twin-width of a collection of graphs [23], obtained via SAT encodings. Hliněný and Jedelský [9] give an upper bound of 8 on the twin-width of planar graphs that almost match the lower bound of 7 obtained by Král' and Lamaison [10]. And graphs with genus g have twin-width $O(\sqrt{g})$ [24].

In this section we will present two works about bounds on twin-width. First we will show that graphs obtained by subdividing at least $2\log n$ times each edge of an n-vertex graph have twin-width at most 4. Secondly, we will show that there are graphs whose twin-width is exponential in treewidth. This matches a result from Jacob and Pilipczuk showing that for every graph G, $\mathrm{tww}(G) \leqslant 3 \cdot 2^{\mathrm{tw}(G)-1}$ [11], where $\mathrm{tw}(G)$ denotes the treewidth of G.

1.2.1 Long subdivisions have twin-width at most four

A subdivision (also known as an edge-subdivision) of a graph G is obtained by replacing each edge e of G by a new vertex linked by two new edges to both endpoints of the edge e.

An ($\geqslant s$)-subdivision (resp. s-subdivision) of a graph G is obtained by subdividing every edge of G at least s times (resp. exactly s times).

In [5], it is proved that the $\Omega(\log n)$-subdivision of any n-vertex graph has bounded twin-width. The proof is rather involved, relies on a characterization by *mixed minors* established in [4], and does not give an explicit constant bound. Here we give an elementary proof that any ($\geqslant 2\lceil \log n \rceil - 1$)-subdivision of an n-vertex graph has twin-width at most 4.

This result has been recently matched with a lower bound by Ahn, Chakraborti, Hendrey, Oum [25]. They show that there are graphs such that every subdivision of them has twin-width at least 4. In the same paper, they also show that every grid larger than 7×7 has twin width exactly 4. This makes graphs of twin-width at most 3 considerably simpler than those of twin-width at most 4, especially among sparse graphs.

Theorem 4. *Let G be a trigraph obtained by subdividing each edge of an n-vertex graph H at least $2\lceil \log n \rceil - 1$ times, and by turning red any subset of its edges as long as the red degree of G remains at most 4, and no vertex with red degree 4 has a black neighbor. Then $tww(G) \leqslant 4$.*

Proof. By no more than doubling the number of vertices of H, we can assume that n is a power of 2. Indeed, padding H with isolated vertices up to the next power of 2 does not change the quantity $\lceil \log |V(H)| \rceil$.

Let G' be a supertrigraph of G obtained by arbitrarily arranging the vertices of H (in G) at the leaves of a "virtual" full binary tree of height $\log n$. So that the red degree does not exceed 4, we so far omit the edges of the tree incident to a leaf (i.e., a vertex of H), while we put in red all the other edges of the tree. (The missing edges of the tree will naturally appear in red.) The internal nodes of the tree are all fresh vertices, not present in G. See figs. 1.2 and 1.3 for an illustration. We show that $\mathrm{tww}(G') \leqslant 4$, hence by Observation 1, $\mathrm{tww}(G) \leqslant 4$ since G is an induced subtrigraph of G'.

We label $1, 2, \ldots, n$ the vertices of H. If there is an edge $ij \in E(H)$, it is subdivided into a path, say, $i, s(ij,1), s(ij,2), \ldots, s(ij,z), j$ in G with $z \geqslant 2\log n - 1$. First, we repeatedly contract adjacent vertices in the middle of this path until it consists of exactly $2\log n$ edges. If $z > 2\log n - 1$, we had to contract at least one pair of adjacent vertices. Thus the vertex in the middle of the path necessarily has now two red edges incident to it. Note that the other edges of the path can be black or red indifferently. To avoid cumbersome notations, we rename the inner vertices of the path $s(ij,1), s(ij,2), \ldots, s(ij,z)$ with now $z = 2\log n - 1$.

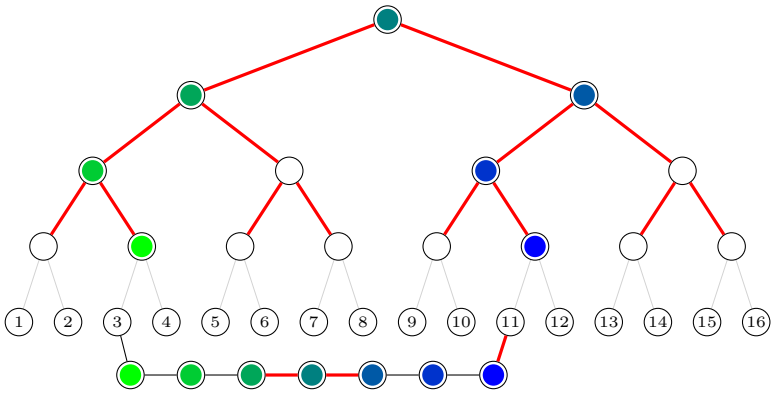


Figure 1.2: Contracting the pairs of vertices with the same color, from the greenest to the bluest, is a partial 4-sequence, which acts as a deletion of the subdivided edge $(3, 11)$.

▷ Claim 5. There is a partial 4-sequence from G' to $G' - \{s(ij,1),\dots,s(ij,z)\}$.

Proof. Intuitively we "zip" the subdivision of ij with the walk made by the union of the path from leaf i to the root, and the path from the root to leaf j. Let $i, v_1, v_2, \dots, v_z, j$ be the concatenation of the simple path from i to the root of the tree, and the simple path from the root to j. Its length is thus $2\log n = z+1$. For h going from 1 to $z = 2\log n - 1$, we contract v_h and $s(ij,h)$ (see fig. 1.2). After each contraction, the newly formed vertex has red degree at most 4. The red degree of vertices that are neither the new vertex nor a leaf of the tree is either unchanged or at most 2. The red degree of a leaf ℓ of the tree may increase by 1. This may only happen the first time a neighbor of ℓ is involved in a contraction, and that contraction merges a black neighbor of ℓ with the parent of ℓ in the tree (like is the case for leaf 3 from fig. 1.2 to fig. 1.3). By assumption, this implies that ℓ had red degree at most 3, thus its red degree does not exceed 4. Thus, what we defined is indeed a partial 4-sequence. One can finally notice that after these z contractions, we indeed reach trigraph $G' - \{s(ij,1),\dots,s(ij,z)\}$. ◁

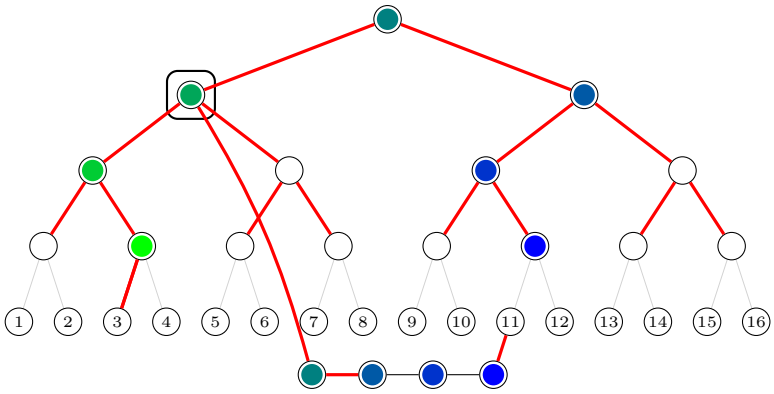


Figure 1.3: The picture after the first three contractions. The newly formed vertex has red degree 4.

We apply claim 5 for each edge of H (or rather, subdivided edge in G). We are then left with a red full binary tree which admits a 3-sequence by lemma 3. Hence there is a 4-sequence for G', and in particular, for G. □

By subdividing each edge two times more, we can simplify the claim. Let's denote $\Delta_R(G)$, the maximum red degree of a vertex in G.

Theorem 6. *Let G be a trigraph obtained by subdividing each edge of an n-vertex graph H at least $2\lceil\log n\rceil + 1$ times, and by turning red any subset of its edges. Then $tww(G) \leqslant \max(4, \Delta_R(G))$.*

Proof. As in the previous proof, we add a virtual full binary tree on the vertices of H. But this time, the virtual edge above each leaf is subdivided once. This means the depth of the tree is increased by one, and that above each leaf there is a vertex of degree 2 in the virtual tree. We add a black edge between each such vertex and its corresponding leaf.

We then contract each edge of H on the red tree as in claim 5. As the depth is increased by 1, it requires each edge to be subdivided two times more, which means $2\lceil\log n\rceil + 1$ times.

Now let us look at the red degree of the vertices of H (the leaves) before all edges of H are contracted. Let v be a vertex of H, its neighbors will be contracted with the parent

of v in the tree. But unlike the previous construction, the parent of v in the tree is only contracted with neighbors of v, or with red neighbors of v.

By definition, the only two way to increase the red degree of v is to contract v with an other vertex or to contract one of its neighbor with one of its non-neighbor. The first one doesn't happen by construction and the second one is ruled out by the previous paragraph. Thus the red degree of v can only decreases.

Therefore the red degree of vertices of the contraction sequence is bounded by $\Delta_R(G)$ for the vertices of H, and the degree of the other vertices is bounded by 4. □

In chapter 2, we will use the following consequence.

Lemma 7. *Let G be a trigraph obtained by subdividing at least $2\lceil \log n \rceil - 1$ times each edge of an n-vertex graph H of degree at most 4, and by turning red all its edges. Then $tww(G) \leqslant 4$.*

1.2.2 Twin width can be exponential in treewidth

Since the beginning of twin-width, it has been known that it generalizes treewidth. In [4], using a very general argument, it is showed that twin-width is bounded by a double exponential of treewidth. More recently, Jacob and Pilipczuk show that for every graph G, $\mathrm{tww}(G) \leqslant 3 \cdot 2^{\mathrm{tw}(G)-1}$ [11].

Conversely, one may ask the following.

▷ Question 1. What is the largest twin-width a graph of treewidth k can have?

A lower bound of $\Omega(k)$ comes from the existence of n-vertex graphs with twin-width $\Omega(n)$ (since the treewidth is trivially upperbounded by $n-1$). This is almost surely the case of graphs drawn from $G(n, 1/2)$, or of the n-vertex Paley graph (for a prime n such that $n \equiv 1 \mod 4$), which has precisely twin-width $(n-1)/2$ [26]. Another example to derive the linear lower bound is the power set graph [11]. Improving on this lower bound is not obvious, and $\Theta(k)$ is indeed the answer to question 1 when replacing 'treewidth' by 'cliquewidth' or 'pathwidth,' or within the class of planar graphs [11].

When switching 'twin-width' and 'treewidth' in question 1, the gap is basically as large as possible: There are examples of n-vertex graphs with treewidth $\Omega(n)$ and twin-width at most 6, in the iterated 2-lifts of K_4 [5, 27].

An important characterization of bounded twin-width is via the absence of complex divisions of an adjacency matrix. A matrix has a *k-mixed minor* if its row (resp. column) set can be partitioned into k sets of consecutive rows (resp. columns), such that each of the k^2 cells defined by this *k-division* contains at least two distinct rows and at least two distinct columns. The *mixed number of a matrix M* is the largest integer k such that M admits a k-mixed minor. The *mixed number of a graph G*, denoted by $\mathrm{mxn}(G)$, is the minimum, taken among all the adjacency matrices M of G, of the mixed number of M. The following was shown.

Theorem 8 ([4])**.** *For every graph G, $(mxn(G)-1)/2 \leqslant tww(G) \leqslant 2^{2^{O(mxn(G))}}$.*

In sparse graphs (here, excluding a fixed $K_{t,t}$ as a subgraph), the previous equivalence is both simpler to formulate and has a better dependency. A matrix has a *k-grid minor* if it has a *k-division* with at least one 1-entry in each of its k^2 cells. The *grid number* of a matrix and of a graph G, denoted by $\mathrm{gn}(G)$, are defined analogously to the previous paragraph. We only state the inequality that is useful to bound the twin-width of a sparse class, but is valid in general.

Theorem 9 (follows from [4]). *For every graph G, $tww(G) \leqslant 2^{O(gn(G))}$.*

Theorems 8 and 9 allow to bound the twin-width of a class $\mathcal{C}$ by exhibiting, for every $G \in \mathcal{C}$, an adjacency matrix of G without large mixed or grid minor. Therefore one merely has to order $V(G)$ (the vertex set of G) in an appropriate way. The double (resp. simple) exponential dependency in mixed number (resp. grid number) implies relatively weak twin-width upper bounds. For several classes whose twin-width was originally upperbounded via theorem 8, better bounds were later given by avoiding this theorem (see [5, 21, 11, 6, 28]). Still for some geometric graph classes, bypassing theorem 8 seems complicated (see for instance [13]). And in general (since this theorem is at the basis of several other applications, see for instance [5, 14, 8]) it would help to have an improved upper bound of $\mathrm{tww}(G)$.

▷ Question 2. Is twin-width sometimes exponential in mixed and grid number?

A variant of twin-width, called *oriented twin-width*, adds an orientation to the red edges (see [29]). The red edge (arc) is oriented away from the contracted vertex. The *oriented twin-width* d of a graph G, denoted by $\mathrm{otww}(G)$, is then defined similarly as twin-width by tolerating more than d red arcs incident to a vertex, as long as at most d of them are outgoing. Rather surprisingly twin-width and oriented twin-width are tied.

Theorem 10 ([29]). *For every graph G, $otww(G) \leqslant tww(G) \leqslant 2^{2^{O(otww(G))}}$.*

It is easy to show that planar graphs have oriented twin-width at most 9. Thus it would be appreciable to lower the dependency of $\mathrm{tww}(G)$ in $\mathrm{otww}(G)$.

▷ Question 3. Is twin-width sometimes exponential in oriented twin-width?

An elementary argument shows that when adding an apex (i.e., an additional vertex with an arbitrary neighborhood) to a graph G, the twin-width of the obtained graph is at most $2\mathrm{tww}(G) + 1$. Again it is not clear whether this increase could be made smaller.

▷ Question 4. Does twin-width sometimes essentially double when an apex is added?

Note that question 1 is asked by Jacob and Pilipczuk [11], and question 3 is posed by Bonnet et al. [29], and is closely related to question 2.

Our contribution. With a single construction, we answer all these questions. The answer to Questions 2, 3, and 4 is 'yes', while the answer to question 1 is $2^{\Theta(k)}$, which confirms the intuition of the authors of [11]. More precisely, we show the following.

Theorem 11. *For every real $0 < \varepsilon \leqslant 1/2$ and positive integer t, there is a graph $G_{t,\varepsilon}$ with a feedback vertex set of size t and such that $tww(G_{t,\varepsilon}) > 2^{(1-\varepsilon)t}$.*

The graph $G_{t,\varepsilon}$ has in particular treewidth at most $t+1$, grid number at most $t+2$, and oriented twin-width at most $t+1$. Thus

- $\mathrm{tww}(G_{t,\varepsilon}) > 2^{(1-\varepsilon)(\mathrm{tw}(G_{t,\varepsilon})-1)}$,
- $\mathrm{tww}(G_{t,\varepsilon}) > 2^{(1-\varepsilon)(\mathrm{gn}(G_{t,\varepsilon})-2)}$, and
- $\mathrm{tww}(G_{t,\varepsilon}) > 2^{(1-\varepsilon)(\mathrm{otww}(G_{t,\varepsilon})-1)}$.

Hence theorem 11 has the following consequences.

Corollary 12. *For every small $\varepsilon > 0$, there is a family $\mathcal{F}$ of graphs with unbounded twin-width such that for every $G \in \mathcal{F}$: $tww(G) > 2^{(1-\varepsilon)(tw(G)-1)}$.*

Up to multiplicative factors, this matches the known upper bound [11, 4], and essentially settles question 1.

Corollary 13. *For every small $\varepsilon > 0$, there is a family $\mathcal{F}$ of graphs with unbounded twin-width such that for every $G \in \mathcal{F}$: $tww(G) > 2^{(1-\varepsilon)(gn(G)-2)}$.*

This answers question 2.

Corollary 14. *For every small $\varepsilon > 0$, there is a family $\mathcal{F}$ of graphs with unbounded twin-width such that for every $G \in \mathcal{F}$: $tww(G) > 2^{(1-\varepsilon)(otww(G)-1)}$.*

This answers question 3.

Corollary 15. *For every small $\varepsilon > 0$, there is a family $\mathcal{F}$ of graphs with unbounded twin-width such that for every $G \in \mathcal{F}$: $tww(G) > (2-\varepsilon)tww(G-\{v\})$, where v is a single vertex of G.*

This answers question 4.

Proof of theorem 11

We fix once and for all, $0 < \varepsilon \leqslant 1/2$, a possibly arbitrarily small positive real. We build for every integer $t > 1/\varepsilon$, the graph $G_{t,\varepsilon}$, that we shorten to G_t, as follows. We set

$$f(t) = \left\lceil 2 + C_t 2^{(1-\varepsilon)t(2+C_t(2^{(1-\varepsilon)t}+1))} \right\rceil$$

where $C_t = 2^{(1-\varepsilon)t}/\varepsilon$.

Construction of G_t. Let T be the full 2^t-ary tree of depth $f(t)$, i.e., with root-to-leaf paths on $f(t)$ edges. Let X be a set of t vertices, that we may identify to $[t]$. The vertex set of G_t is $X \uplus V(T)$. The edges of G_t are such that $G[X]$ is an independent set, and $G[V(T)] = T$. The edges between $V(T)$ are X are such that

- the root of T has no neighbor in X, and
- the 2^t children (in T) of every internal node of T each have a distinct neighborhood in X.

Note that this defines a single graph up to isomorphism. By a slight abuse of language, we may utilize the usual vocabulary on trees directly on G_t. By *root*, *internal node*, *child*, *parent*, *leaf* of G_t, we mean the equivalent in T.

We start with this straightforward observation.

Lemma 16. *G_t has treewidth at most $t+1$.*

Proof. The set X is a feedback vertex set of G_t of size t, thus $\mathrm{tw}(G_t) \leqslant \mathrm{fvs}(G_t) + 1 \leqslant t+1$. □

The following is the core lemma, which occupies us for the remainder of the section.

Lemma 17. *G_t has twin-width greater than $2^{(1-\varepsilon)t}$.*

Proof. We assume, by way of contradiction, that G_t admits a d-sequence with $d \leqslant 2^{(1-\varepsilon)t}$. We consider the partial d-sequence $\mathcal{S}$, starting at G_t, and ending right before the first contraction involving a child of the root. We first show that no vertex of X can be involved in a contraction of $\mathcal{S}$. Note that it implies, in particular, that the root cannot be involved in a contraction of $\mathcal{S}$.

▷ Claim 18. No part of $\mathcal{S}$ contains more than one vertex of X.

Proof of the claim. Observe that, for every $i \neq j \in [t]$, there are 2^{t-1} sets of $2^{[t]}$ containing exactly one of i, j: 2^{t-2} only contains i, and 2^{t-2} only contains j. Recall now that by assumption, in every trigraph of $\mathcal{S}$, every child of the root is alone in its part. Thus a part P of $\mathcal{S}$ such that $|P \cap X| \geqslant 2$ would have red degree at least $2^{t-1} > 2^{(1-\varepsilon)t} \geqslant d$. □

▷ Claim 19. No part of $\mathcal{S}$ intersects both X and $V(T)$.

Proof of the claim. For the sake of contradiction, consider the first occurrence of a part $P \supseteq \{x, v\}$ with $x \in X$ and $v \in V(T)$. Vertex x is adjacent to half of the children of the root, whereas v is adjacent to at most one of them, or all of them (if v is itself the root). In both cases, this entails at least $2^{t-1} - 1$ red edges for P towards children of the root. If v is not a grandchild of the root, the red degree of P is at least 2^{t-1}. We thus assume that v is a grandchild of the root.

As $t \geqslant 2$, there is a $y \in X \setminus \{x\}$. Let v' be the child of v whose neighborhood in X is exactly $\{y\}$. This vertex exists since $f(t) \geqslant 3$. If P contains v', P is also red-adjacent to $\{y\}$ (indeed a part, by claim 18). If instead, P does not contain v', then P is also red-adjacent to the part containing v'.

Thus, in any case, the red degree of P is at least $2^{t-1} > 2^{(1-\varepsilon)t} \geqslant d$. □

From Claims 18 and 19, we immediately obtain:

▷ Claim 20. Every part of $\mathcal{S}$ intersecting X is a singleton.

Crucial to the proof, we introduce two properties $\mathscr{P}$, and later $\mathscr{Q}$, on internal nodes $v \in V(T)$ in trigraphs $H \in \mathcal{S}$. Property $\mathscr{P}$ is defined by

$$\mathscr{P}(v, H) = \text{“At least } 2^{\varepsilon t} \text{ children of } v \text{ are in the same part of } \mathcal{P}(H).\text{”}$$

We first remark that any internal node in a non-singleton part verifies $\mathscr{P}$.

▷ Claim 21. Let H be any trigraph of $\mathcal{S}$ and v be any internal node of T whose part in $\mathcal{P}(H)$ is not a singleton. Then $\mathscr{P}(v, H)$ holds.

Proof of the claim. Let P be the part of v in $\mathcal{P}(H)$, and $u \in P \setminus \{v\}$. At least $2^t - 1$ children of v are not adjacent to u. Thus these $2^t - 1$ vertices have to be in at most $d + 1 \leqslant 2^{(1-\varepsilon)t} + 1$ parts. These parts are part P, plus at most d parts linked to P by a red edge. Since $(2^{\varepsilon t} - 1)(2^{(1-\varepsilon)t} + 1) < 2^t - 1$ (recall that $\varepsilon < 1/2$), one of these parts (possibly P) contains at least $2^{\varepsilon t}$ children of v. □

As the merge of a singleton part $\{v\}$ with any other part does not change the intersections of parts with the set of children of v, we get a slightly stronger claim.

▷ Claim 22. Let v be an internal node of T, and H be the last trigraph of $\mathcal{S}$ for which v is in a singleton part of $\mathcal{P}(H)$. Then $\mathscr{P}(v, H)$ holds.

A *preleaf* is an internal node of T adjacent to a leaf, i.e., the parent of some leaves. We obtain the following as a direct consequence of claim 21.

▷ Claim 23. In any trigraph $H \in \mathcal{S}$, any non-preleaf internal node $v \in V(T)$ that verifies $\mathscr{P}(v, H)$ has at least $2^{\varepsilon t}$ children u verifying $\mathscr{P}(u, H)$.

We define the property $\mathscr{Q}$ on internal nodes v of T and trigraphs $H \in \mathcal{S}$ by induction:

$$\mathscr{Q}(v,H) = \begin{cases} \mathscr{P}(v,H) & \text{if } v \text{ is a preleaf, and otherwise} \\ \mathscr{Q}(u_1,H) \wedge \mathscr{Q}(u_2,H) & \text{for some pair } u_1 \neq u_2 \text{ of children of } v. \end{cases}$$

That is, $\mathscr{Q}$ is defined as $\mathscr{P}$ for preleaves, and otherwise, $\mathscr{Q}$ holds when it holds for at least two of its children. Observe that $\mathscr{P}$ and $\mathscr{Q}$ are monotone in the following sense: If $\mathscr{P}(v,H)$ (resp. $\mathscr{Q}(v,H)$) holds, then $\mathscr{P}(v,H')$ (resp. $\mathscr{Q}(v,H')$) holds for every subsequent trigraph H' of the partial d-sequence $\mathcal{S}$. We may write that v *satisfies* $\mathscr{P}$ *(resp.* $\mathscr{Q}$*) in* H when $\mathscr{P}(v,H)$ (resp. $\mathscr{Q}(v,H)$) holds, and may add *for the first time* if no trigraph $H' \in \mathcal{S}$ before H is such that $\mathscr{P}(v,H')$ (resp. $\mathscr{Q}(v,H')$) holds.

▷ Claim 24. For any trigraph $H \in \mathcal{S}$ and internal node v of T, $\mathscr{P}(v,H)$ implies $\mathscr{Q}(v,H)$.

Proof of the claim. This is a tautology if v is a preleaf. The induction step is ensured by claim 23, since $2^{\varepsilon t} \geqslant 2$. □

At the end of the partial d-sequence $\mathcal{S}$, we know, by claim 22, that at least one child of the root satisfies $\mathscr{P}$, hence satisfies $\mathscr{Q}$, by claim 24. Thus the first time in the partial d-sequence $\mathcal{S}$ that $\mathscr{Q}(v,H)$ holds, for a trigraph $H \in \mathcal{S}$ and a child v of the root, is well-defined. We call F this trigraph, and v_0 a child of the root satisfying $\mathscr{Q}(v_0,F)$.

We now find many nodes satisfying $\mathscr{Q}$ in F, whose parents form a vertical path of singleton parts.

▷ Claim 25. There is a set $Q \subset V(T)$ of at least $f(t)-2$ internal nodes such that
- for every $v \in Q$, $\mathscr{Q}(v,F)$ holds,
- the parent of any $v \in Q$ is in a singleton part of $\mathcal{P}(F)$, and
- and *no* two distinct nodes of Q are in an ancestor-descendant relationship.

Proof of the claim. We construct by recurrence two sequences $(v_i)_{i\in[f(t)-2]}, (q_i)_{i\in[0,f(t)-3]}$ of internal nodes of T such that for all $i \in [f(t)-2]$, v_i is a child of v_{i-1}, v_{i-1} is in a singleton part of $\mathcal{P}(F)$, and v_{i-1} has a child $q_{i-1} \neq v_i$ satisfying $\mathscr{Q}(q_{i-1},F)$.

Assume that the sequence is defined up to v_i, for some $i < f(t)-2$. We will maintain the additional invariant that v_i satisfies $\mathscr{Q}$ for the first time in F. This is the case for $i = 0$.

As v_i is not a preleaf, it satisfies $\mathscr{Q}$ for the first time when a second child satisfies $\mathscr{Q}$. Let v_{i+1} be this second child, and q_i be the first child to satisfy $\mathscr{Q}$ (breaking ties arbitrarily if both children satisfy $\mathscr{Q}$ for the first time in F). The vertex v_{i+1} satisfies $\mathscr{Q}$ for the first time in F. Thus our invariant is preserved.

For every $i \in [f(t)-2]$, v_i is in a singleton part of $\mathcal{P}(F)$. Indeed, by claim 22, if v_i was not in a singleton part of $\mathcal{P}(F)$, v_i would satisfy $\mathscr{P}$, hence $\mathscr{Q}$, in the trigraph preceding F; a contradiction.

The set Q can thus be defined as $\{q_i \,:\, i \in [0, f(t)-3]\}$. We already checked that the first two requirements of the lemma are fulfilled. No pair in Q is in an ancestor-descendant relationship since the nodes of Q are all children of a root-to-leaf path made by the v_is (see fig. 1.4). □

Let B the vertices $w \in V(F)$ such that $w(G)$ contains at least $2^{\varepsilon t}$ children of the same node of T. Each vertex of B is red-adjacent to at least $\log(2^{\varepsilon t}) = \varepsilon t$ (singleton) parts of X. Therefore, since the red degree of (singleton) parts of X is at most $2^{(1-\varepsilon)t}$:

$$|B| \leqslant \frac{2^{(1-\varepsilon)t}}{\varepsilon}.$$

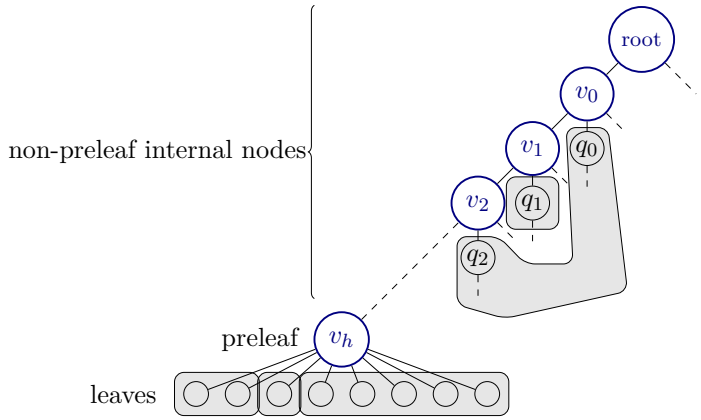


Figure 1.4: The nodes $(v_i)_{i\in[0,h]}$ and $(q_i)_{[0,h-1]}$ ($h = f(t) - 2$) satisfy $\mathscr{P}$ and $\mathscr{Q}$ in F. The v_is and the root (nodes circled in blue) are in singleton parts of F. The other represented nodes can be in larger parts (shaded areas).

Next we show that there is relatively large set of vertices of F each corresponding to a non-singleton part and containing an internal node of T.

▷ Claim 26. There is a set $B' \subseteq V(F)$ of size at least

$$\frac{1}{(1-\varepsilon)t} \log\left(\frac{f(t)-2}{|B|}\right) - 1$$

such that for every $b \in B'$ there is an internal node v of T with $v \in b(G_t)$ and $|b(G_t)| \geqslant 2$.

Proof of the claim. Let $s := \frac{1}{(1-\varepsilon)t} \log(\frac{f(t)-2}{|B|}) - 1$. Our goal is to construct a sequence $(b_i)_{i\in[0,s]}$ of distinct vertices of F such that for every $i \in [s]$,

$$\text{part } b_i(G_t) \text{ is } \textit{not} \text{ a singleton and contains an internal node of } T. \tag{1.1}$$

We first focus on finding b_0. Note that b_0 need not satisfy Invariant (1.1), but will be chosen to force the existence of b_1 itself satisfying (1.1) and starting the induction.

Let $Q := \{q_j \ : \ 0 \leqslant j \leqslant f(t) - 3\} \subset V(T)$ be as described in claim 25. Every $q_j \in Q$ has (at least) one descendant q'_j that is a preleaf and satisfies $\mathscr{Q}$, hence $\mathscr{P}$, in F. The q'_js are pairwise distinct because no two nodes of Q are in an ancestor-descendant relationship. We set $Q' := \{q'_j \ : \ 0 \leqslant j \leqslant f(t) - 3\}$.

Now for every q'_j, at least $2^{\varepsilon t}$ of its children are in the same part of $\mathcal{P}(F)$; hence, this part corresponds to a vertex in B. By the pigeonhole principle, there is a $b_0 \in B$ that contains at least $2^{\varepsilon t}$ children of at least $(f(t) - 2)/|B|$ nodes of Q'.

For each b_i, we define $Q_i \subset Q$ as the set of vertices q_j such that

- $b_i(G_t)$ contains a (not necessarily strict) descendant z of q_j, and
- no part $b_{i'}(G_t)$ with $i' < i$ contains a node on the path between q_j and z in T.

Thus $|Q_0| \geqslant (f(t) - 2)/|B|$.

We now assume that $b_i \in V(F)$, for some $0 \leqslant i < s$, has been found with

$$|Q_i| \geqslant \frac{f(t)-2}{|B| \cdot 2^{i(1-\varepsilon)t}}. \tag{1.2}$$

Observe that Q_0 satisfies (1.2). We construct b_{i+1}, Q_{i+1} satisfying the invariants (1.1) and (1.2).

For each $q_j \in Q_i$, consider the highest descendant z_j of q_j in $b_i(G_t)$, and z'_j the parent of z_j in T. By construction, the part P_j of $\mathcal{P}(F)$ containing z'_j is not a $b_k(G_t)$ for any $k \leqslant i$. Part P_j is linked to $b_i(G_t)$ by a red edge. Therefore there are at most $2^{(1-\varepsilon)t}$ such parts P_j. In particular, there is a $b_{i+1} \in V(F)$ such that $b_{i+1}(G_t)$ contains at least

$$\frac{|Q_i|}{d} \geqslant \frac{f(t)-2}{|B| \cdot 2^{i(1-\varepsilon)t}} \cdot \frac{1}{2^{(1-\varepsilon)t}} = \frac{f(t)-2}{|B| \cdot 2^{(i+1)(1-\varepsilon)t}}$$

parents z'_j of highest descendants z_j.

Remark that $b_{i+1}(G_t)$ has size at least two while $(f(t)-2)/(|B| \cdot 2^{(i+1)(1-\varepsilon)t}) > 1$, which holds since $i < s$. Thus $b_{i+1}(G_t)$ does not contain any parent v_j of a q_j (since the v_js are in singleton parts). In particular, $|Q_{i+1}| \geqslant (f(t)-2)/(|B| \cdot 2^{(i+1)(1-\varepsilon)t})$, and b_{i+1}, Q_{i+1} satisfy (1.1) and (1.2).

Finally, the set $B' := \{b_i \ : \ 1 \leqslant i \leqslant s\}$ has the required properties. □

We can now finish the proof of the lemma.

For every $b_i \in B'$, let $u_i \in b_i(G_t)$ be an internal node of T. As $b_i(G_t) \geqslant 2$, u_i satisfies $\mathscr{P}$ in F. This implies that b_i or a red neighbor of b_i is in B. Therefore, the total number of red edges incident to a vertex of B is at least $|B'| - |B|$. Thus there is a vertex in B with red degree at least $(|B'| - |B|)/|B|$. This is a contradiction since

$$\begin{aligned}
\frac{|B'|-|B|}{|B|} = \frac{|B'|}{|B|} - 1 &\geqslant \left(\frac{1}{(1-\varepsilon)t} \log\left(\frac{f(t)-2}{|B|}\right) - 1\right) \cdot \frac{1}{|B|} - 1 \\
&\geqslant \left(\frac{1}{(1-\varepsilon)t} \log\left(2^{(1-\varepsilon)t(2+C_t \cdot (2^{(1-\varepsilon)t}+1))}\right) - 1\right) \cdot \frac{1}{|B|} - 1 \\
= \left((2 + C_t \cdot (2^{(1-\varepsilon)t}+1)) - 1\right) \cdot \frac{1}{|B|} - 1 &> 2^{(1-\varepsilon)t} + 1 - 1 = 2^{(1-\varepsilon)t} \geqslant d.
\end{aligned}$$

since, we recall, $f(t) = \left\lceil 2 + C_t \cdot 2^{(1-\varepsilon)t(2+C_t \cdot (2^{(1-\varepsilon)t}+1))} \right\rceil$ and $C_t = \dfrac{2^{(1-\varepsilon)t}}{\varepsilon} \geqslant |B|$. □

Since X is a feedback vertex set of size t of G_t, lemma 17 implies theorem 11, and hence corollary 12.

As the twin-width of T is 2, adding the t apices in X, multiplies the twin-width by at least $2^{t(1-\varepsilon-\frac{1}{t})}$. Thus one apex in X multiplies the twin-width by at least $2^{1-\varepsilon-\frac{1}{t}}$, which can be made arbitrarily close to 2. This establishes corollary 15.

Oriented twin-width and grid number

In this section, we check that G_t has oriented twin-width at most $t+1$, and grid number at most $t+2$.

A *(partial) oriented contraction sequence* is defined similarly as a *(partial) contraction sequence* with every red edge replaced by a red arc leaving the newly contracted vertex. Then a *(partial) oriented d-sequence* is such that all the vertices of all its trigraphs have at most d out-going red arcs. The *oriented twin-width* of a graph G, denoted by $\mathrm{otww}(G)$, is the minimum integer d such that G admits an oriented d-sequence.

Lemma 27. *The oriented twin-width of G_t is at most $t+1$.*

Proof. We observe that the 2-sequence for trees [4] is an oriented 1-sequence. We contract T to a single vertex (without touching X) in that manner. This yields a partial oriented $t+1$-sequence for G_t ending on a $t+1$-vertex trigraph, which can be contracted in any way. This contraction sequence witnesses that $\mathrm{otww}(G_t) \leqslant t+1$. □

Thus corollary 14 holds.

We finish by establishing corollary 13.

Lemma 28. *The grid number of G_t is at most $t + 2$.*

Proof. Recall that $V(G_t) = X \uplus V(T)$. Let $\prec$ be the total order on $V(G_t)$ that puts first all the vertices of X in any order, then from left to right, all the leaves of T, followed by the preleaves, the nodes at depth $f(t) - 2$, the nodes at depth $f(t) - 3$, and so on, up to the root. We denote by M the adjacency matrix of G_t ordered by $\prec$.

Let M_T be the submatrix of M obtained by deleting the t rows and t columns corresponding to X. Note that the grid number of M is at most $\mathrm{gn}(M_T) + t$. We claim that there is no 3-grid minor in M_T.

Indeed, in the order $\prec$, above the diagonal of M_T there is no pair of 1-entries in strictly decreasing positions. Thus overall there is no triple of 1-entries in strictly decreasing positions. Thus no 3-grid minor is possible. □

Chapter 2

Deciding twin-width at most 4 is NP-hard

Contents

In this chapter, we show that determining if an n-vertex graph has twin-width at most 4 is NP-complete, and requires time $2^{\Omega(n/\log n)}$ unless the Exponential-Time Hypothesis fails. Along the way, we also show how to encode trigraphs H (2-edge colored graphs involved in the definition of twin-width) into graphs G, in the sense that every d-sequence (sequence of vertex contractions witnessing that the twin-width is at most d) of G inevitably creates H as an induced subtrigraph, whereas there exists a partial d-sequence that actually goes from G to H.

2.1 Introduction

One of the main algorithmic interests with twin-width is that first-order (FO) model checking, that is, deciding if a first-order sentence φ holds in a graph G, can be decided in fixed-parameter time (FPT) $f(|\varphi|, d) \cdot |V(G)|$ for some computable function f, when given a d-sequence of G [4]. As for most classes known to have bounded twin-width, one can compute $O(1)$-sequences in polynomial time for members of the class, the latter result unifies and extends several known results [30, 31, 32, 33, 3] for hereditary (but not necessarily monotone) classes.

For monotone (i.e., subgraph-closed) classes, the FPT algorithm of Grohe, Kreutzer, and Siebertz [34] for FO model checking on nowhere dense classes, is complemented by W[1]-hardness on classes that are somewhere dense (i.e., *not* nowhere dense) [35], and even AW[∗]-hardness on classes that are *effectively* somewhere dense [36]. The latter results mean that, for monotone classes, FO model checking is unlikely to be FPT beyond nowhere dense classes.

The missing piece for an FO model-checking algorithm in FPT time on any class of bounded twin-width is a polynomial-time algorithm and a computable function f, that given a constant integer bound c and a graph G, either finds an $f(c)$-sequence for G, or correctly reports that the twin-width of G is greater than c. The running time of the algorithm could be $n^{g(c)}$, for some function g. However to get an FPT algorithm in the combined parameter *size of the sentence + bound on the twin-width*, one would further require that the approximation algorithm takes FPT time in c (now seen as a parameter), i.e., $g(c)n^{O(1)}$. We know such an algorithm for instance on ordered graphs (more generally, ordered binary structures) [8], graphs of bounded clique-width, proper minor-closed classes [4], but not in general graphs.

On the other hand, prior to this work, no algorithmic lower bound was known for computing the twin-width. Our main result rules out an (exact) XP algorithm to decide $tww(G) \leqslant k$, that is, an algorithm running in time $n^{f(k)}$ for some computable function f. Indeed we show that deciding if the twin-width of a graph is at most 4 is intractable. We refer the reader to section 2.1.2 for some context on the Exponential-Time Hypothesis (ETH), which implies that n-variable 3-SAT cannot be solved in time $2^{o(n)}$.

Theorem 29. *Deciding if a graph has twin-width at most 4 is NP-complete. Furthermore, no algorithm running in time $2^{o(n/\log n)}$ can decide if an n-vertex graph has twin-width at most 4, unless the ETH fails.*

As far as approximation algorithms are concerned, our result only rules out a ratio better than 5/4 for determining the twin-width. This still leaves plenty of room for an $f(\mathrm{OPT})$-approximation, which would be good enough for most of the (theoretical) algorithmic applications. Note that such algorithms exist for treewidth in polytime [37] and FPT time [38], for pathwidth [37], and for clique-width via rank-width [39].

Is theorem 29 surprising? On the one hand, it had to be expected that deciding, given a graph G and an integer k, whether $\mathrm{tww}(G) \leqslant k$ would be NP-complete. This is the case for example of treewidth [40], pathwidth [41, 42, 43], clique-width [44], rank-width [45], mim-width [46], and bandwidth [47]. On the other hand, the parameterized complexity of these width parameters is more diverse and harder to predict. Famously, Bodlaender's algorithm is a linear FPT algorithm to exactly compute treewidth [2] (and a non-uniform FPT algorithm came from the Graph Minor series [48]). In contrast, while there is an XP algorithm to compute bandwidth [49], an FPT algorithm is highly unlikely [50]. It is a long-standing open whether an FPT or a mere XP algorithm exist for computing clique-width exactly, or even simply if one can recognize graphs of clique-width at most 4 in polynomial time (deciding clique-width at most 3 is indeed tractable [51]).

Theorem 29 almost completely resolves the parameterized complexity of exactly computing twin-width on general graphs. Two questions remain: can graphs of twin-width at most 2, respectively at most 3, be recognized in polynomial time. Graphs of twin-width 0 are cographs, which can be recognized in linear time [52], while it was recently shown that graphs of twin-width at most 1 can be recognized in polynomial time [17].

Contrary to the hardness proof for treewidth [40], which involves some structural characterizations by chordal completions, and the intermediate problems Minimum Cut

Linear Arrangement, Max Cut, and Max 2-SAT [47], our reduction is "direct" from 3-SAT. This makes the proven hardness of twin-width more robust, and easier to extend to restricted classes of graphs, especially sparse ones. Theorem 29 holds for bounded-degree input graphs. For instance, performing our reduction from Planar 3-SAT produces subgraphs of constant powers of the planar grid (while admittedly weakening the ETH lower bound from $2^{\Omega(n/\log n)}$ to $2^{\Omega(\sqrt{n}/\log n)}$). Hence, while the complexity status of computing treewidth on planar graphs is a famous long-standing open question, one can probably extend the NP-hardness of *twin-width at most 4* to planar graphs, by tuning and/or replacing the few non-planar gadgets of our reduction.

Let us point out that, in contrast to subset problems, there is no $2^{O(n)}$-time algorithm known to compute twin-width. The exhaustive search takes time $n^{2n+O(1)}$ by considering all sequences of $n-1$ pairs of vertices. We leave as an open question whether the ETH lower bound of computing twin-width can be brought from $2^{\Omega(n/\log n)}$ to $2^{\Omega(n)}$, or even $2^{\Omega(n\log n)}$. The latter lower bound is known to hold for Subgraph Isomorphism [53] (precisely, given a graph H and an n-vertex graph G, deciding if H is isomorphic to a subgraph of G requires time $2^{\Omega(n\log n)}$, unless the ETH fails), or computing the Hadwiger number [54] (i.e., the size of the largest clique minor).

2.1.1 Outline of the proof of theorem 29

We propose a quasilinear reduction from 3-SAT. Given an n-variable instance I of 3-SAT, we shall construct an $O(n\log n)$-vertex graph $G = G(I)$ which has twin-width at most 4 if and only if I is satisfiable.

Half of our task is to ensure that no 4-sequence will exist if I is unsatisfiable. This is challenging since many contraction strategies are to be considered and addressed. We make this task more tractable by attaching *fence gadgets* to some chosen vertex subsets. The effect of the fence *enclosing* S is that no contraction can involve vertices in S with vertices outside of S, while S is not contracted into a single vertex. The *maximal or outermost* fences (we may nest two or more fence gadgets) partition the rest of the vertices. This significantly tames the potential 4-sequences of G.

Our basic building block, the *vertical set*, consists of a pair of vertices (*vertical pair*) enclosed by a fence. It can be thought of as a bit set to 0 as long as the pair is not contracted, and to 1 when the pair gets contracted. It is easy to assemble vertical sets as prescribed by an auxiliary digraph D (of maximum degree 3), in such a way that, to contract (by a partial 4-sequence) the pair of a vertical set V, one first has to contract all the vertical sets that can reach V in D. This allows to propagate and duplicate a bit in a so-called *wire* (corresponding to an out-tree in D), and to perform the logical AND of two bits.

The bit propagation originates from a variable gadget (we naturally have one per variable appearing in I) that offers two alternatives. One can contract the "top half" of the gadget of variable x_i, which then lets one contract the vertical sets in the wire of literal x_i, or one can contract instead the "bottom half" of the gadget, as well as the vertical sets in the wire of literal $\neg x_i$. Concretely, these two contraction schemes represent the two possible assignments for variable x_i. A special "lock" on the variable gadget (called *half-guards*) prevents its complete contraction, and in particular, performing contractions in both the wires of a literal and its negation.

The leaves of the literal wires serve as inputs for 3-clause gadgets. One can contract the output (also a vertical set) of a clause gadget if and only if one of its input is previously contracted. We then progressively make the AND of the clauses via a "path" of binary AND gadgets fed by the clause outputs. We eventually get a vertical set, called *global*

output, which can be contracted by a partial 4-sequence only if I is satisfiable. Indeed at this point, the variable gadgets are still locked so at most one of their literals can be propagated. This ticks one of our objective off. We should now ensure that a 4-sequence is possible from there, when I is satisfiable.

For that purpose, we add a wire from the global output back to the half-guards (or locks) of the variable gadgets. One can contract the vertical sets of that wire, and in particular the half-guards. Once the variable gadgets are "unlocked," they can be fully contracted. As a consequence, one can next contract the wires of literals set to false, and *all* the remaining vertical sets involved in clause gadgets.

At this point, the current trigraph H roughly has one vertex per outermost fence with red edges linking two adjacent gadgets (and no black edge). We will guarantee that the (red) degree of H is at most 4, its number of vertices of degree at least 3 is at most βn, for some constant β. Besides we will separate gadgets by degree-2 wires of length $2\log(\beta n)$ beforehand. This is crucial so that the red graph of H is a $(2\log n')$-subdivision of an n'-vertex graph. We have indeed showed that such trigraphs have twin-width at most 4.

This finishes to describe our overall plan for the reduction and its correctness. It happens that fence gadgets are easier to build as trigraphs, while the rest of the gadgetry can be directly encoded by graphs. We thus show how to encode trigraphs by graphs, as follows. For any trigraph J whose red graph has degree at most d, and component size at most h, there is a graph G on at most $f(d,h)\cdot|V(J)|$ vertices such that J has twin-width at most $2d$ if and only if G has twin-width at most $2d$. This uses some local replacements and confluence properties of certain partial contraction sequences.

2.1.2 The Exponential-Time Hypothesis

The Exponential-Time Hypothesis (ETH, for short) was proposed by Impagliazzo and Paturi [55] and asserts that there is no subexponential-time algorithm solving 3-SAT. More precisely, there is an $\varepsilon > 0$ such that n-variable 3-SAT cannot be solved in time $2^{\varepsilon n}$. Impagliazzo et al. [56] present a subexponential-time Turing-reduction parameterized by a positive real $\varepsilon > 0$ that, given an n-variable m-clause k-SAT-instance I, produces at most $2^{\varepsilon n}$ k-SAT-instances $I_1, \ldots, I_t$ such that I is satisfiable if and only if at least one of $I_1, \ldots, I_t$ is satisfiable, each I_i having no more than n variables and $C_\varepsilon n$ clauses for some constant C_ε depending only on ε. This crucial reduction is known as the Sparsification Lemma. Due to it, there is an $\varepsilon' > 0$ such that there is no algorithm solving m-clause 3-SAT in time $2^{\varepsilon' m}$, unless the ETH fails.

A classic reduction from Tovey [57], linear in the number of clauses, then shows the following.

Theorem 30 ([57, 56])**.** *The n-variable* 3-SAT *problem where each variable appears at most twice positively, and at most twice negatively, is NP-complete, and cannot be solved in time $2^{o(n)}$, unless the ETH fails.*

2.1.3 Organization of this chapter

The rest of the chapter is organized as follows. In section 2.2 we present how to encode trigraphs into graphs. In section 2.3, we start by quickly recapping the overall plan. The following subsections go through the various gadgets. Finally section 2.3.7 details the quasilinear reduction from 3-SAT to the problem of deciding if the twin-width is at most 4, and its correctness.

2.2 Encoding a trigraph by a graph

In this subsection, we present a construction allowing to encode trigraphs into (plain) graphs. Our objective is, given a trigraph H with red degree at most d, to produce a graph G such that H admits a $2d$-sequence iff G admits a $2d$-sequence.

Formally, we build a graph G for which every $2d$-sequence $\mathcal{S}$ inevitably creates H as an induced subtrigraph of a trigraph of $\mathcal{S}$. By Observation 1, the existence of a $2d$-sequence for G implies the existence of one for H. Moreover, this construction is such that there is a partial $2d$-sequence which, from G, exactly reaches H. Therefore, any $2d$-sequence for H can be augmented to become a $2d$-sequence for G.

At first sight, our construction works only for trigraphs H with a connected red graph. We could show the following.

Lemma 31. *For every trigraph H such that $\mathcal{R}(H)$ is connected and of degree at most d, there is a graph G such that:*

- *every $2d$-sequence of G goes through a supertrigraph of H, and*
- *there is a partial $2d$-sequence from G to H.*

However, we will need a stronger version, allowing to simulate trigraphs with disconnected red graphs. First we show how to turn the trigraph induced by one connected component of the red graph into a plain graph (this is lemma 32). Later we loop through all connected components of the red graph with size at least 2, and apply lemma 32 (that is done in lemma 36).

Lemma 32. *Let H be a trigraph, and $S \subseteq V(H)$ be the vertex set of a connected component of $\mathcal{R}(H)$ with degree at most d. There is a trigraph G and a set $T \subseteq V(G)$ satisfying the following statements:*

1. *$|T|$ is bounded by a function of d and $|S|$ only,*
2. *no red edge touches T (in particular, $G[T]$ has no red edge),*
3. *$G - T$ is isomorphic to $H - S$,*
4. *there is a partial $2d$-sequence from G to H, and*
5. *every $2d$-sequence of G goes through a supertrigraph of some trigraph $\tilde{H}$, where $\tilde{H}$ can be obtained from H by performing contractions* not *involving vertices of S, nor creating red edges incident to S.*

Proof. We begin with the construction of G. The vertex set of G can be split into two sets: on one hand $V(H) \setminus S$, and on the other hand T, which will be defined in function of $H[S]$. Let $\{v_1, v_2, \ldots, v_{|S|}\}$ be the set S. To build set T, we blow up every vertex v_i of S into a copy L_i of the biclique $K_{t,t}$ with $t = 2(2d+2) \cdot (2d)^{|S|-1} + 1$. The two *sides* of each biclique L_i are denoted by $A_i = \{a_{i,j}\}_{j\in[t]}$ and $B_i = \{b_{i,j}\}_{j\in[t]}$, and are respectively called *A-side* and *B-side*. The vertex set $T \subseteq V(G)$ is:

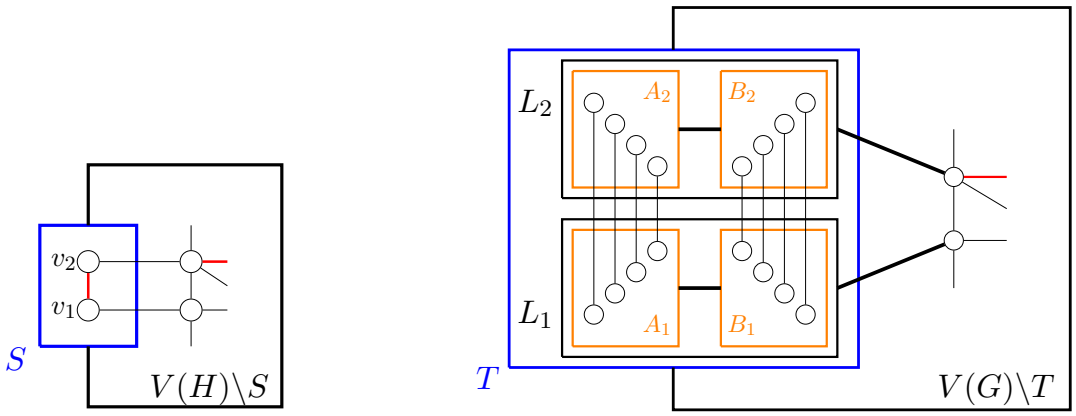
$$T = \bigcup_{v_i \in S} V(L_i) = \bigcup_{v_i \in S, j \in [t]} \{a_{i,j}, b_{i,j}\}.$$

In addition to the edges forming each biclique, we replace every black edge of $H[S]$ by a complete bipartite graph between the two corresponding bicliques. Furthermore, every red edge of $H[S]$ becomes a black canonical matching, with edges $a_{i,j}a_{i',j}$ and $b_{i,j}b_{i',j}$. Formally,

$$E(G[T]) = \bigcup_{v_i \in S} A_i \times B_i \;\cup \bigcup_{v_i v_{i'} \in E(H)} L_i \times L_{i'} \;\cup \bigcup_{\substack{v_i v_{i'} \in R(H) \\ j \in [t]}} \{a_{i,j}a_{i',j}, b_{i,j}b_{i',j}\}.$$

We finish the construction by considering the black edges initially connecting S and $V(H) \setminus S$ in the trigraph H. For any edge $v_i z \in E(H)$ with $v_i \in S$ and $z \in V(H) \setminus S$, we add a black edge between any vertex of the biclique L_i and $z \in V(G) \setminus T$. In summary,

$$E(G) = E(G[T]) \cup E(H - S) \cup \bigcup_{\substack{v_i \in S \\ z \in V(H) \setminus S \\ v_i z \in E(H)}} L_i \times \{z\}.$$



(a) An example of trigraph H. Set S consists of two vertices v_1 and v_2, linked by a red edge. Vertices of $H - S$ adjacent to S are also represented.

(b) Graph G obtained from H. Thick black edges link every vertex of one endpoint to every vertex of the other endpoint. Set T contains two bicliques L_1 and L_2, inherited from $S = \{v_1, v_2\}$.

Figure 2.1: An example of encoding: induced trigraph S (left) and the plain graph T (right).

Statements 1-3. The first three statements are satisfied by this construction. In particular the size of T is $2t|S|$, and t was defined as a function of d and $|S|$. Moreover, no vertex of $V(H) \setminus S$ was modified by this construction, so $G - T$ is isomorphic to $H - S$.

Statement 4. We focus now on the fourth statement: We exhibit a partial $2d$-sequence $\mathcal{S}^*$ from G to H. Let us begin with a short description of $\mathcal{S}^*$. This particular sequence never contracts two vertices of T lying in different bicliques. Moreover, after a contraction of two vertices belonging to the same biclique, it forces the same contraction in all other bicliques of T. We now describe $\mathcal{S}^*$ in detail.

We start by contracting $a_{i,1}$ and $a_{i,2}$ for each $v_i \in S$. Then, we contract $\{a_{i,1}, a_{i,2}\}$ with $a_{i,3}$ for each $v_i \in S$, and so on. When the A-side of each biclique L_i is contracted into a single vertex $\{a_{i,1}, a_{i,2}, \ldots, a_{i,t}\}$, we proceed similarly with the B-side. Finally, each biclique L_i contains exactly two contracted vertices, which are respectively A_i and B_i. We contract all these pairs in each L_i: every biclique is now contracted into a single vertex.

The obtained trigraph is isomorphic to H. Indeed, if $v_i v_{i'}$ is a black edge (resp. a non-edge) in $H[S]$, then L_i and $L_{i'}$ are homogeneous and, as contracted vertices, they are connected with a black edge (resp. a non-edge). However, if $v_i v_{i'}$ is a red edge in $H[S]$, then there is a semi-induced matching connecting bicliques L_i and $L_{i'}$ in $G[T]$, so the contracted vertices L_i and $L_{i'}$ are linked by a red edge. Finally, the black edges between S and $V(H) \setminus S$ are preserved because the contractions of $\mathcal{S}^*$ occur inside bicliques of T which are homogeneous (i.e., fully adjacent or fully non-adjacent) with every vertex $z \in V(H) \setminus S$.

Let us check that the partial sequence $\mathcal{S}^*$ only goes through trigraphs with red degree at most $2d$. As we only contract vertices within the same side of each biclique first, no

red edge appears between two vertices of the same biclique. Moreover, if $v_iv_{i'} \notin R(H)$, then bicliques L_i and $L_{i'}$ are homogeneous, so no red edge appears between two vertices belonging respectively to L_i and $L_{i'}$. Thus, the red edges may only bridge two different bicliques $L_i, L_{i'}$ such that $v_iv_{i'} \in R(H)$. The way we progressively contract the matching between L_i and $L_{i'}$, a contracted vertex within L_i is the endpoint of at most two red edges towards $L_{i'}$. Assume without loss of generality that $u(G) = \{a_{i,1}, a_{i,2}, \ldots, a_{i,j+1}\}$ is contracted in A_i while only $u'(G) = \{a_{i',1}, a_{i',2}, \ldots, a_{i',j}\}$ is contracted in $A_{i'}$ for some $j \in [t-1]$. There are two red edges between bicliques L_i and $L_{i'}$ which are uu' and uu'', where $u''(G) = \{a_{i',j+1}\}$. As the red degree of $H[S]$ is at most d, we indeed have that $\mathcal{S}^*$ is a partial $2d$-sequence.

Statement 5. We terminate by showing that any $2d$-sequence $\mathcal{S}$ from graph G necessarily produces at some moment a supertrigraph of $\tilde{H}$, where $\tilde{H}$ is obtained from H with contractions not involving set S. As this part of the proof is more intricate than the previous ones, two intermediate claims are stated. We fix $g(d,t) = (t-1)/(2d+2) = 2(2d)^{|S|-1}$.

▷ Claim 33. Let G_h be any trigraph of $\mathcal{S} = G, \ldots, G_h, \ldots, K_1$. If a vertex u of G_h is such that $u(G)$ intersects two distinct bicliques L_i and $L_{i'}$, then there is a vertex w (possibly u) of G_h intersecting one side of L_i in at least $g(d,t)$ elements: $|w(G) \cap A_i| \geqslant g(d,t)$ or $|w(G) \cap B_i| \geqslant g(d,t)$.

Proof. Let $a \in u(G) \cap V(L_i)$ and $z \in u(G) \cap V(L_{i'})$. We assume w.l.o.g. that the partite set of the biclique L_i in which a lies is A_i. We identify a set $X \subset V(G)$ of size at least $t-1$ such that, for any vertex $u' \in V(G_h) \setminus \{u\}$ such that $u'(G) \cap X \neq \emptyset$, there is a red edge uu'. The composition of set X depends on the adjacency between v_i and $v_{i'}$ in H.

- If $v_iv_{i'}$ is a non-edge in H, then we fix $X = B_i$. Indeed, any element of B_i is adjacent to a but not to z, so there is a red edge between u and any contracted vertex u' intersecting B_i. Moreover, $|B_i| = t$.
- If $v_iv_{i'}$ is a black edge in H, then we fix $X = A_i \setminus \{a\}$. Any element of $A_i \setminus \{a\}$ is not adjacent to a but adjacent to z. Thus, any contracted vertex u intersecting $A_i \setminus \{a\}$ is connected to u in red. Moreover, $|A_i \setminus \{a\}| = t-1$.
- If $v_iv_{i'}$ is a red edge in H, then we fix $X = B_i \setminus \{z'\}$, where z' is the matching neighbor of z in L_i. Observe that z' may belong to A_i and, in this case, $X = B_i$. Any element of X is adjacent to a but not to z. Moreover, $|X| \in \{t-1, t\}$.

In summary, there is a set X of at least $t-1$ vertices such that any contracted vertex $u' \in G_h$ intersecting X is linked by a red edge to u. Consequently, there cannot be more than $2d+1$ contracted vertices $w \in G_h$ intersecting X, otherwise the red degree of u is necessarily larger than $2d$. For this reason, at least one vertex $w \in G_h$ which intersects X has size at least $\lceil (t-1)/(2d+1) \rceil \geqslant g(d,t)$. In each case, either $X \subsetneq A_i$ or $X \subseteq B_i$, so either $|w(G) \cap A_i| \geqslant g(d,t)$ or $|w(G) \cap B_i| \geqslant g(d,t)$. ◁

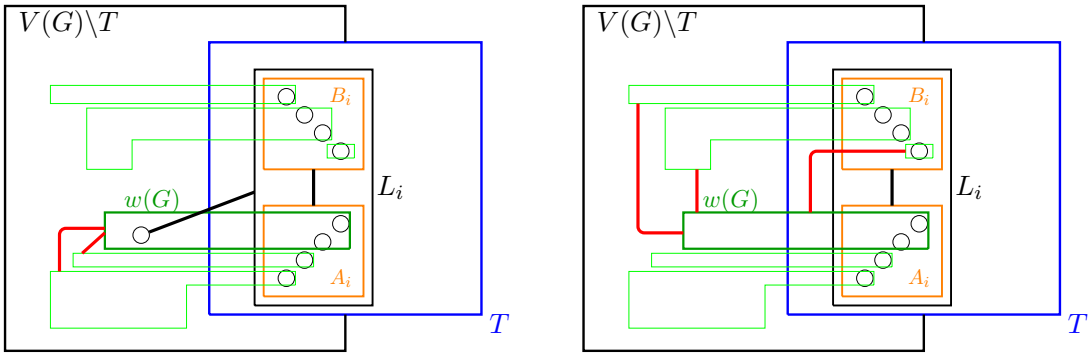
Let G' be the first trigraph of sequence $\mathcal{S}$ having a vertex u such that $|u(G) \cap T| \geqslant g(d,t)$. We denote by G'' the trigraph preceding G' in $\mathcal{S}$. We show that every contracted vertex $w \in G'$ is either fully contained in some biclique L_i or contained in $V(G) \setminus T$.

▷ Claim 34. Every contracted vertex $u \in V(G')$ satisfies:

- either $u(G) \subseteq A_i$ or $u(G) \subseteq B_i$ for some L_i of T,
- or $u(G) \subseteq V(G) \setminus T$.

Furthermore, there is no red edge between a contracted vertex inside a biclique of T and a contracted vertex in $G - T$.

Proof. We begin by proving that if a contracted vertex $w \in V(G')$ intersects T, then it intersects at most one of its bicliques, that is, $w(G) \cap T \subseteq L_i$ for some index i. According



(a) Case when v_i is adjacent to some vertex of $w(G) \setminus A_i$ in H.

(b) Case when v_i is not adjacent to any vertex of $w(G) \setminus A_i$ in H

Figure 2.2: How contractions intersecting both T and $V(G) \setminus T$ would imply large red degree in G'. The green boxes represent contracted vertices in G'.

to claim 33, all the contracted vertices of G'' intersect T within a single biclique. This holds in particular for the two vertices, say $x, y \in V(G'')$, whose contraction yields u. Assume that x and y are both contained within different bicliques, say, L_i and $L_{i'}$, respectively. By claim 33, there is a vertex of G' intersecting L_i in at least $g(d,t)$ elements. By definition of trigraph G', this vertex is necessarily u. This yields a contradiction, as the cardinality of $x(G) = u(G) \cap V(L_i)$ would be at least $g(d,t)$. Therefore, vertex u intersects at most one biclique. In brief, from now on, every set $w(G)$ with $w \in V(G')$ can be written as the union of elements of some biclique L_i with vertices of $G - T$.

Each side, A_i or B_i, of a biclique L_i satisfies the following property: There are at least $2d+2$ contracted vertices of G' intersecting every A_i (or B_i). Any contracted vertex in G'' covers less than $g(d,t)$ elements of T, by definition. So, at least $2d+3$ contracted vertices of G'' must intersect A_i as $|A_i| = t$. In G', this property holds except for vertex u, which is the contraction of two vertices $x, y \in V(G'')$. Therefore, as announced, at least $2d+2$ contracted vertices w of G' satisfy $w(G) \cap A_i \neq \emptyset$ (resp. $w(G) \cap B_i \neq \emptyset$) for every side A_i (resp. B_i).

Based on this property, we show that, for any $w \in V(G')$, set $w(G)$ cannot contain both vertices of T and of $V(G) \setminus T$. Assume it is the case and w.l.o.g. that $w(G) \cap A_i \neq \emptyset$. All vertices of L_i play the same role regarding the adjacencies between T and $G-T$: either they are all connected to some $z \in G - T$ (if $v_i z \in E(H)$) or none of them is connected to z (if $v_i z \notin E(H)$). Consequently, we can distinguish only two cases. First, assume that at least one vertex of $w(G) \setminus A_i$ is adjacent (black or red) to L_i in G (Figure 2.2a). In that case, w admits at least $2d+1$ red neighbors in G' which are the other contracted vertices intersecting A_i. Indeed, these vertices contain elements of A_i which are adjacent to $w(G) \setminus A_i$ but not to $w(G) \cap A_i$. Second, assume that no edge of G connects $w(G) \setminus A_i$ with L_i (Figure 2.2b). We can identify $2d+1$ red neighbors of w: among the at least $2d+2$ contracted vertices of G' intersecting B_i, at least $2d+1$ of them are different from w. These contracted vertices contain elements of B_i which are adjacent to $w(G) \cap A_i$ but not to $w(G) \setminus A_i$. In summary, for any $w \in G'$, either $w(G) \subseteq L_i$ or $w(G) \subseteq V(G) \setminus T$.

We now prove that there is no red edge $ww' \in R(G')$ such that $w(G) \subseteq L_i$ and $w'(G) \subseteq V(G) \setminus T$. Suppose by way of contradiction that such an edge exists. As all vertices of L_i play the same role in the adjacencies between T and $G - T$, there is a red edge between w' and any vertex of G' intersecting L_i. As at least $2d+2$ of these vertices intersect A_i, the red degree of w' is at least $2d+2$, a contradiction.

To end the proof, we show that any contracted vertex $w(G) \subseteq L_i$ verifies either $w(G) \subseteq A_i$ or $w(G) \subseteq B_i$. Consider the at least $2d+1$ vertices of G' intersecting A_i different from w. They contain elements of A_i which are adjacent to $w(G) \cap A_i$ but not to $w(G) \cap B_i$. Therefore, the red degree of w is at least $2d+1$, a contradiction. $\lhd$

In brief, the partial sequence from G to G' does not use contractions mixing vertices of T and vertices of $V(G) \setminus T$. As a consequence, G' can be split in two parts: its induced subtrigraph G'_T obtained from contractions on the bicliques and its induced subtrigraph G'_{G-T} obtained from the contractions on $G-T$. Our objective is to prove that H appears in the first part.

We now focus more specifically on the vertex u. W.l.o.g., we assume that $u(G)$ intersects $A_i \subset L_i$. We denote by J_0 a set of exactly $g(d,t)$ distinct indices j such that $a_{i,j} \in u(G)$. For every $J \subseteq [t]$, we denote by $A_{i,J}$ the set $\{a_{i,j} : j \in J\}$. It holds that $|J_0| = |A_{i,J_0}| = g(d,t)$.

▷ Claim 35. *$H[S]$ is an induced subtrigraph of G'_T.*

Proof. We initialize a subset $Y \subseteq S$ to $\{v_i\}$, an injective mapping $\rho : Y \hookrightarrow V(G')$ to $v_i \mapsto u$, and a subset $J \subseteq [t]$ to J_0. We maintain the following invariant:

$$(*)\ A_{h,J} \subseteq \rho(v_h)(G) \text{ for every } v_h \in Y, \text{ and } 2 \leqslant |J| = \frac{t-1}{(2d+2)\cdot(2d)^{|Y|-1}}.$$

The invariant initially holds by construction of J_0. While $Y \neq S$, we pick a pair $v_h \in Y, v_{h'} \in S \setminus Y$ such that $v_h v_{h'} \in R(H)$. Since $v_h(G) \supseteq A_{h,J}$ and there is an induced matching between $A_{h,J}$ and $A_{h',J}$, the set $A_{h',J}$ can be spanned by at most $2d$ parts of $\mathcal{P}(G')$. Thus we select of vertex $w_{h'} \in V(G')$ and a subset $J' \subseteq J$ of size $\frac{|J|}{2d}$ such that $w_{h'}(G) \supseteq A_{h',J'}$. We set the new J to J', and we augment ρ with $v_{h'} \mapsto w_{h'}$. Finally, we add $v_{h'}$ to Y.

The invariant $(*)$ is preserved by construction. Since the graph $\mathcal{R}(H)$ is connected, this process ends when $Y = S$ and $|J| = \frac{g(d,t)}{(2d)^{|S|-1}} = 2$. We claim that trigraph $G'[\rho(S)]$, which is an induced subtrigraph of G'_T, is isomorphic to $H[S]$. Indeed, let $v_h, v_{h'} \in S$ be any pair of vertices, and let $w_h = \rho(v_h)$ and $w_{h'} = \rho(v_{h'})$. Assume first that $v_h v_{h'} \in R(H)$. As $w_h(G) \supseteq A_{h,J}$, $w_{h'}(G) \supseteq A_{h',J}$, $|J| = 2$, and there is an induced matching between $A_{h,J}$ and $A_{h',J}$, there is a red edge in G' between w_h and $w_{h'}$. If $v_h v_{h'}$ is a black edge in $H[S]$, then any pair of vertices in $w_h(G) \times w_{h'}(G)$ is a black edge in G. Hence, sets $w_h(G)$ and $w_{h'}(G)$ are homogeneous and $w_h w_{h'}$ is a black edge in G'. Eventually, if $v_h v_{h'}$ is a non-edge in $H[S]$, both $w_h(G)$ and $w_{h'}(G)$ are homogeneous in the sense that they are not adjacent at all. So, $w_h w_{h'}$ is a non-edge in G'. In brief, $G'[\rho(S)]$ is isomorphic to $H[S]$. $\lhd$

This concludes the proof of lemma 32. Indeed, consider the subtrigraph of G' induced on vertices of both sets $\rho(S)$ and G'_{G-T}. The subtrigraph $G'[\rho(S)]$ is isomorphic to $H[S]$ (claim 35) and the second part G'_{G-T} is obtained from contractions on $G-T$ which do not make red edges appear towards S (claim 34). □

If the red graph $\mathcal{R}(H)$ is not connected, one can apply lemma 32 for each of its connected components.

Lemma 36. *Given any trigraph H whose red graph has degree at most d and connected components of size at most h, one can compute in time $O_{d,h}(|V(H)|)$ a graph G on $O_{d,h}(|V(H)|)$ vertices such that H has a $2d$-sequence if and only if G has a $2d$-sequence.*

Proof. Let $S_1, \ldots, S_r$ be the connected components of $\mathcal{R}(H)$ of size at least 2. We define a collection of trigraphs $H_0 = H, H_1, \ldots, H_r$, where H_r is a plain graph isomorphic to G. Each trigraph $H_{\ell+1}$ will be built from H_ℓ by applying lemma 32 on $S_{\ell+1}$. We proceed by induction and prove, for any $0 \leqslant \ell \leqslant r$, that:

- $|V(H_\ell)|$ is linear in $|V(H)|$ if $d, h = O(1)$: $|V(H_\ell)| = f_\ell(d,h)|V(H)|$,
- the connected components of $\mathcal{R}(H_\ell)$ of size at least 2 have vertex sets $S_{\ell+1}, \ldots, S_r$,
- H_ℓ has a $2d$-sequence iff H has a $2d$-sequence.

The base case is trivial, as $H_0 = H$. Let us assume, for the induction step, that H_ℓ satisfies these three properties. We consider the red component $S_{\ell+1}$ in H_ℓ to build trigraph $H_{\ell+1}$. We apply lemma 32 with trigraph H_ℓ and vertex set $S_{\ell+1}$. The trigraph $H_{\ell+1}$ substitutes $H_\ell[S_{\ell+1}]$ with a plain graph $H_{\ell+1}[T]$ not touched by a red edge. Thus the vertex sets of connected components of $\mathcal{R}(H_{\ell+1})$ of size at least 2 are those of $\mathcal{R}(H_\ell)$ minus $S_{\ell+1}$.

Moreover, the size of T is bounded by d and $|S_{\ell+1}|$ only. As a consequence,

$$|V(H_{\ell+1})| = |T| + |V(H_\ell) \setminus S_{\ell+1}| \leqslant h(d, |S_{\ell+1}|) + f_\ell(d,h)|V(H)| \leqslant f_{\ell+1}(d,h)|V(H)|,$$

where h is the function defined accordingly to lemma 32.

We finally show that $H_{\ell+1}$ has a $2d$-sequence iff H_ℓ has a $2d$-sequence. This is a direct consequence of lemma 32. On one hand, there is a $2d$-partial sequence from $H_{\ell+1}$ to H_ℓ, so if H_ℓ has a $2d$-sequence, then $H_{\ell+1}$ has one, too. On the other hand, if $H_{\ell+1}$ has a $2d$-sequence, then one trigraph of the sequence is a supertrigraph of $\tilde{H}_\ell$, where $\tilde{H}_\ell$ is a trigraph which can be obtained from H_ℓ by performing contractions preserving S_ℓ as a red connected component of the trigraph. One can deduce from this statement a $2d$-sequence of H_ℓ: first contract H_ℓ to obtain $\tilde{H}_\ell$, then conclude by Observation 1.

As a conclusion, trigraph $G = H_r$ satisfies the induction hypotheses: its red graph is edgeless and it has a $2d$-sequence if and only if H admits one, too. □

In the sequel, we will use lemma 36 with $d = 2$ and $h = 12$. In particular, the size of the encoding graph will be linear in the trigraph. As paths are connected graphs of degree at most 2, we will invoke this scaled-down version.

Lemma 37. *Given any trigraph H whose red graph is a disjoint union of 12-vertex paths and isolated vertices, one can compute in polynomial time a graph G on $O(|V(H)|)$ vertices such that H has twin-width at most 4 if and only if G has twin-width at most 4.*

2.3 Hardness of determining if the twin-width is at most four

Here we show the main result of the chapter.

Theorem 29. *Deciding if a graph has twin-width at most 4 is NP-complete. Furthermore, no algorithm running in time $2^{o(n/\log n)}$ can decide if an n-vertex graph has twin-width at most 4, unless the ETH fails.*

The membership to NP is ensured by the d-sequence: a polynomial-sized certificate that a graph has twin-width at most d, checkable in polynomial time. We thus focus on the hardness part of the statement, and design a quasilinear reduction from 3-SAT.

Using the result of section 2.2, our task is now slightly simpler. Given a 3-SAT instance I, we may design a *trigraph* satisfying the requirements of lemma 37 with twin-width at most 4 if and only if I is satisfiable.

As already mentioned, given an instance I of 3-SAT we will create an equivalent instance of the problem: *is trigraph G of twin-width at most 4?* We now present the various gadgets used in our reduction. The only building block featuring red edges is the fence gadget. Its red graph is a 12-vertex path that is a connected component in the red graph of the overall construction G. Hence G can be turned into a graph with only a constant multiplicative blow-up, by lemma 37.

The correctness of the reduction naturally splits into two implications:

- (i) If G admits a 4-sequence, then I is satisfiable.
- (ii) If I is satisfiable, then G admits a 4-sequence.

We will motivate all the gadgets along the way, by exhibiting key properties that they impose on a potential 4-sequence. These properties readily lead to a satisfying assignment for I. So the proof of (i) will mostly consist of aggregating lemmas specific to individual gadgets.

We also describe partial 4-sequences to reduce most of the gadgets. However some preconditions (specifying the context in which a particular gadget stands) tend to be technical, and make more sense after the construction of G. In those cases, to avoid unnecessarily lengthy lemmas, we only give an informal strategy, and postpone the adequate contraction sequence to the final proof of (ii).

2.3.1 Fence gadget

We now design a gadget F partitioned into two vertex sets A, B. The gadget is *attached* to a non-empty subset $S \subseteq V(G) \setminus (A \cup B)$ by making A and S fully adjacent, and B and S fully non-adjacent. Our intent is that, in a 4-sequence, a vertex of F can be contracted with another vertex of the graph only when S has been contracted into a single vertex. We will ensure that every vertex outside $S \cup V(F)$ is fully adjacent to B or fully non-adjacent to A. As $|A| = |B| = 6 > 4$ will hold, the effect is that vertices of S have to be contracted to a single vertex before any vertex of S (or subset of vertices within S) can be contracted with a vertex or subset of vertices in $G - S$. To summarize, F encloses S into an "unbreakable unit:" the inside of S cannot be contracted with the outside as long as S is not a single vertex. Hence we call F a *fence gadget*.

The fence gadget is defined as a trigraph whose red graph is a simple path on 12 vertices, in particular, a connected graph of maximum degree 2. In the overall trigraph G, no red edge will link a vertex in the gadget to a vertex outside of it. Thus the fence gadget can be replaced by a graph by lemma 37.

We now define the fence gadget. Its vertex set is $A \cup B$ with $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ and $B = \{b_1, b_2, b_3, b_4, b_5, b_6\}$. Its black edge set consists of 13 edges: the cycles $a_1a_2a_3a_4a_5a_6a_1$ and $b_1b_2b_3b_4b_5b_6b_1$, plus the edge b_1a_6. Its red edge set consists of 11 edges: a_ib_i for each $i \in [6]$, and a_ib_{i+1} for each $i \in [5]$. Finally A is made fully adjacent to S. See fig. 2.3 for an illustration.

We will later nest fence gadgets. Thus we have to tolerate that F has other neighbors than S in G. Actually we even allow $V(F)$ to have neighbors outside of S and the fence gadgets surrounding F. We however always observe the following rule.

Definition 38 (Attachment rule)**.** *A fence gadget F with vertex bipartition (A, B), and attached to S, satisfies the* attachment rule *in a trigraph H if F is a connected component of $\mathcal{R}(H)$, and there is a set $X \subseteq V(H) \setminus (A \cup B \cup S)$ such that:*

- $\forall x \in A$, $N(x) \setminus V(F) = X \cup S$,
- $\forall x \in B$, $N(x) \setminus V(F) = X$, *and*
- $\forall x \in X$, $S \subset N(x)$.

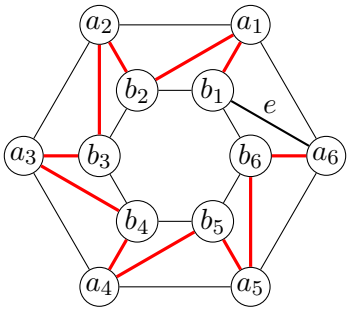


Figure 2.3: The fence gadget F, with $A = \{a_i \mid 1 \leqslant i \leqslant 6\}$ and $B = \{B_i \mid 1 \leqslant i \leqslant 6\}$.

Initially in G, we make sure that all the fence gadgets satisfy the attachment rule. This will remain so until we decide to contract them.

We denote by Y the set $V(G) \setminus (V(F) \cup S \cup X)$, when X is defined in the attachment rule. We make the following observations on the three possible neighborhoods that vertices outside of F have within $V(F)$.

Observation 39. *The fence gadget definition and the attachment rule implies:*

- $\forall x \in S$, *it holds* $N(x) \cap V(F) = A$,
- $\forall x \in X$, *it holds* $N(x) \cap V(F) = V(F) = A \cup B$, *and*
- $\forall x \in Y$, *it holds* $N(x) \cap V(F) = \emptyset$.

Henceforth we will represent every fence gadget as a brown rectangle surrounding the set S it is attached to. The vertices of X are linked to the brown rectangle, as they are fully adjacent to $S \cup V(F)$. See fig. 2.4 for an illustration of the attachment rule, and a compact representation of fence gadgets, and fig. 2.5 for two nested fence gadgets and how the attachment rule is satisfied for both.

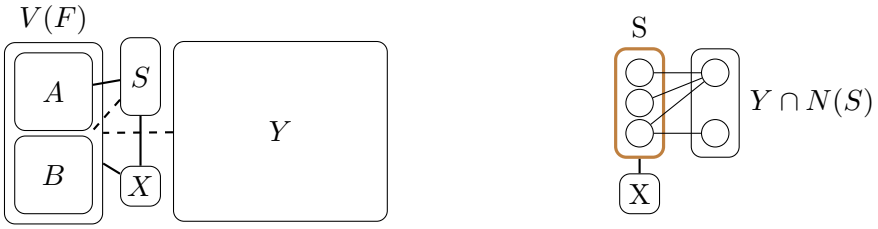


Figure 2.4: Left: The forced adjacencies (solid lines, all edges between the two sets) and non-adjacencies (dashed lines, no edge between the two sets), as specified by the attachment rule. Right: Symbolic representation of the fence gadget attached to S by a brown rectangle; the vertices in X are linked to the brown box, while the potential neighbors of S in Y are only linked individually to their neighbors in S, and are fully non-adjacent to the vertices of the fence $V(F)$. Possible edges between X and Y are *not* represented.

Constraints of the fence gadget on a 4-sequence. The following lemmas are preparatory steps for the milestone that no part in a 4-sequence of G can overlap S (that is, intersects S without containing it).

Lemma 40. *The first contraction involving two vertices of F results in a vertex of red degree at least 5, except if it is a contraction of some $a_i \in A$ with some $b_j \in B$.*

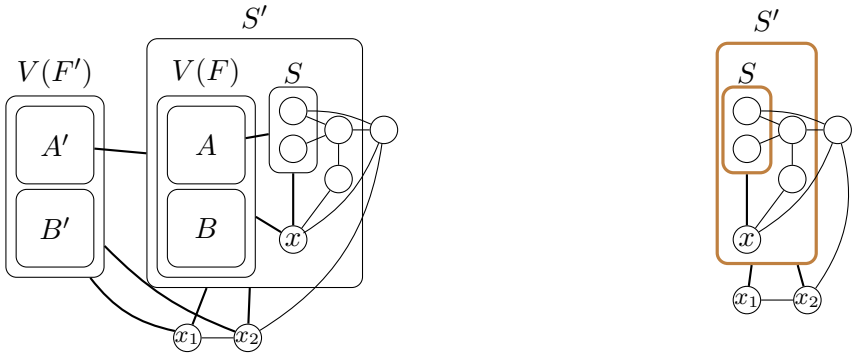


Figure 2.5: Left: Two nested fences, where the set X of the innermost fence is $\{x, x_1, x_2\} \cup V(F')$, while the set X' of the outermost fence is $\{x_1, x_2\}$. Right: Its compact representation.

Proof. We consider a pair of distinct vertices u, v of $A \times A$ or $B \times B$. If u and v are non-adjacent in F, they both have at least three private neighbors in the total graph $\mathcal{T}(F)$. Thus their contraction results in a vertex of red degree at least 6, since by assumption these at least 6 vertices lie in distinct parts.

If u and v are adjacent in F, they both have at least two private neighbors in $\mathcal{T}(F)$, and a common neighbor w such that at least one of uw and vw is red in F. Hence their contraction results in a vertex with red degree at least 5. □

Lemma 41. *If the first contraction involving two vertices of F is of some $a_i \in A$ with some $b_j \in B$, the red degree within F of the created vertex is at least 3.*

Proof. Let $u \in A$ and $v \in B$. Then, in the total graph $\mathcal{T}(F)$, either u and v both have at least two private neighbors, or they both have at least one private neighbor and a common neighbor linked to u or v in red in the trigraph F. By assumption, these at least 4 neighbors or at least 3 neighbors are in distinct parts. Hence, in both cases, the contraction of u and v results in a vertex of red degree at least 3 within F. □

The last preparatory step is this easy lemma.

Lemma 42. *Before a contraction involves two vertices of $V(F)$, the following holds in a partial 4-sequence:*

- *no part intersects both X and S,*
- *no part intersects both Y and S, and*
- *no part intersects both X and Y.*

Proof. By Observation 39, such a part would have red degree $|B| = 6$, $|A| = 6$, and $|A \cup B| = 12$, respectively. □

As a consequence we obtain the following.

Lemma 43. *In a partial 4-sequence of G, the first contraction involving a vertex in $V(F)$ and a vertex in $V(F) \cup S$ has to be done after S is contracted into a single vertex.*

Proof. We consider the first time a vertex $u \in V(F)$ is involved in a contraction with a vertex of $V(F) \cup S$. Either (case 1) the part of u, P_u, is contracted with a part P_v containing $v \in V(F)$, or (case 2) P_u is contracted with a part P intersecting S but not $V(F)$.

In case 1, by lemma 40, u and v hit both A and B. Thus, by lemma 41, the red degree within F of the resulting vertex z is at least 3. Moreover z is linked by a red edge to every part within S, since S is fully adjacent to A, and fully non-adjacent to B. Thus S should at this point consist of a single part.

We now argue that case 2 is impossible in a partial 4-sequence. By lemma 42, part P cannot intersect $X \cup Y$ (nor $V(F)$, by construction). Thus $P \subseteq S$. If $u \in A$, then the contraction of P_u and P has incident red edges toward at least 5 vertices: three vertices of A non-adjacent to u and two private neighbors of u (in the total graph) within B. If instead $u \in B$, the red degree of the contracted part is at least 6, as witnessed by two neighbors of u in B, and four non-neighbors of u in A. □

We can now establish the main lemma on how a fence gadget constrains a 4-sequence. lemmas 42 and 43 have the following announced consequence: While S is not contracted into a single vertex, no part within S can be contracted with a part outside of S, and similarly vertices of X cannot be contracted with vertices of Y.

Lemma 44. *In a partial 4-sequence, while S is not contracted to a single vertex,*

- *(i) no part intersects both S and $V(G) \setminus S$, nor*
- *(ii) both X and Y.*

Proof. Let H be a trigraph obtained by a partial 4-sequence from G, such that S is not contained in a part of $\mathcal{P}(H)$. By lemma 43, no pair of vertices in $V(F) \times (S \cup V(F))$ are in the same part of $\mathcal{P}(H)$ (since S is not contracted to a single vertex). Thus we conclude by lemma 42. □

Contracting the fence gadget. The previous lemmas establish some constraints that the fence gadget imposes on a supposed (partial) 4-sequence. We now see how a partial 4-sequence actually contracts a fence gadget.

Every time we are about to contract a fence gadget F attached to S, we will ensure that the following properties hold:

- no prior contraction has involved a vertex of $V(F)$,
- no red edge has one endpoint in $V(F)$ and one endpoint outside $V(F)$, and
- S is contracted into a single vertex with red degree at most 3.

In particular, the fence gadget F still satisfies the attachment rule.

Lemma 45. *Let H be a trigraph containing a fence gadget F attached to a single vertex s of red degree at most 3. We assume that F respects the attachment rule in H.*

Then there is a partial 4-sequence from H to H', where H' is the trigraph obtained from H by contracting $V(F)$ into a single vertex.

Proof. As F respects the attachment rule in H, every vertex of F has the same (fully black) neighborhood in $V(H) \setminus (V(F) \cup \{s\})$. Thus, contractions within $V(F)$ will only create red edges within $V(F) \cup \{s\}$. We can therefore focus on the trigraph induced by $V(F) \cup \{s\}$.

Recall the vertex labels of fig. 2.3. We first contract a_1 and b_1. This creates a vertex c_1 of red degree 4, and in particular it adds one to the red degree of s, which has now red degree at most 4 (see left-hand side of fig. 2.6). Now we will contract within A, and within B, not to increase more the red degree of s.

Vertices b_2 and b_3 now have only 4 neighbors in total in the fence gadget. Thus we can contract them into b_{23} and keep red degree at most 4 (see middle of fig. 2.6). In turn, a_2 and a_3 can be contracted into a_{23} for the same reason (see right of fig. 2.6). Then we

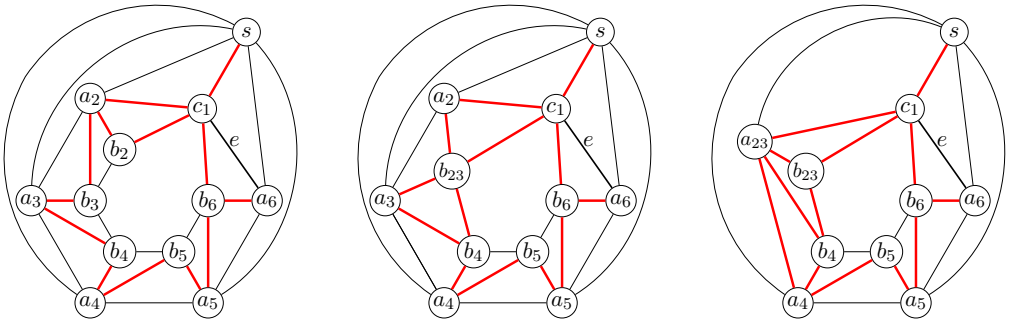


Figure 2.6: The first three contractions of the fence gadget.

contract b_4 with b_{23}, forming b_{234}, and a_4 with a_{23}, forming a_{234}. We contract b_5 and b_{234} into b_{2345}, and a_5 and a_{234} into a_{2345}. We finally contract b_6 and b_{2345} into b, and a_6 and a_{2345} into a. We contract a and c_1, then we contract the resulting vertex and b.

Crucially the edge e stays black until the contraction of a_6 and a_{2345}, so c_1 maintains a red degree of at most 4. Also importantly, the only contractions involving a vertex of A and a vertex of B are the first and last two contractions. Thus s is incident in red to at most one vertex of the fence gadget. □

2.3.2 Propagation, wire, and long chain

A *vertical set* V consists of two vertices x, y combined with a fence gadget F attached to $\{x, y\}$. Thus $V = \{x, y\} \cup V(F)$. We call *vertical pair* the vertices x and y. We will usually add a superscript to identify the different copies of vertical sets: Every vertex whose label is of the form u^j belongs to the vertical set V^j.

The *propagation gadget* from V^j to V^k puts all the edges between V^j and x^k (and no other edge). We call these edges an *arc from* V^j *to* V^k. We also say that the vertical set V^k is *guarded* by V^j. The pair V^j, V^k is said *adjacent*. Here, singleton $\{x^k\}$ plays the role of X (Definition 38) for the attachment rule of vertical set V_j.

The *propagation digraph of* G, denoted by $\mathcal{D}(G)$, has one vertex per vertical set and an arc between two vertical sets linked by an arc (in the previous sense). A (maximal) *wire* W is an induced subgraph of G corresponding in $\mathcal{D}(G)$ to a (maximal) out-tree on at least two vertices. See fig. 2.7 for an illustration of a wire made by simply concatenating propagation gadgets. Eventually $\mathcal{D}(G)$ will have out-degree at most 2, in-degree at most 2, and total degree at most 3.

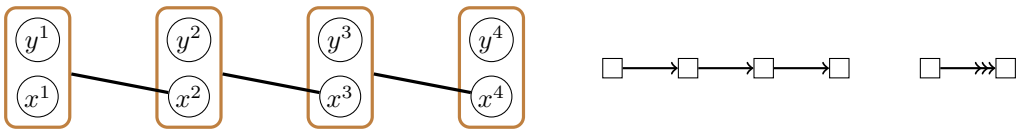


Figure 2.7: Left: A non-branching wire made of 4 vertical sets and 3 propagation gadgets. Every vertical set is guarded by the vertical set just to its left. Center: A more compact representation, which corresponds to the propagation digraph. Right: Symbolic representation of the long chain, that is, of the represented wire if $L = 4$.

The *children* of a vertical set V are the vertical sets that V guards. The *root of wire* W is the unique vertical set of in-degree 0 in $\mathcal{D}(W)$. The *leaves of wire* W are the vertical sets of out-degree 0 in $\mathcal{D}(W)$. A wire is said *primed* when the vertical pair of its root has been contracted.

A wire W is *non-branching* if every vertex of $\mathcal{D}(W)$ has out-degree at most 1; hence, $\mathcal{D}(W)$ is a directed path. A *long chain* is a wire W such that $\mathcal{D}(W)$ is a directed path on L vertices, where integer L will be specified later (and can be thought as logarithmic in the total number of fences which do not belong to vertical sets). Otherwise, if $\mathcal{D}(W)$ has at least one vertex with out-degree at least 2, wire W is said *branching*. A vertical set with two children is also said *branching*. See fig. 2.8 for an example of a branching wire with exactly one branching vertical set.

Constraints of the propagation gadget on a 4-sequence. We provide the proof that a contraction in a vertical set V is only possible when the vertical pair of all the vertical sets V' with a directed path to V in $\mathcal{D}(G)$ has been contracted.

Lemma 46. *Let V^j and V^k be two vertical sets with an arc from V^j to V^k. In a partial 4-sequence from G, any contraction involving two vertices of V^k has to be preceded by the contraction of x^j and y^j.*

Proof. We recall the notations of section 2.3.1. Let F be the fence gadget attached to $S = \{x^j, y^j\}$, X the neighborhood of $V(F)$ outside of $S \cup V(F)$, and Y the vertices that are not in $S \cup V(F) \cup X$. (We always assume that the attachment rule is satisfied.) We have $x^{j'} \in X$ and $y^k \in Y$, therefore by the second item of lemma 44, their contraction has to be preceded by the contraction of x^j and y^j. Now applying the first item of lemma 44 to the fence gadget F' attached to $S' = \{x^k, y^k\}$, and lemma 43, any contraction involving a pair of V^k distinct from S' has to be preceded by the contraction of x^k and y^k. □

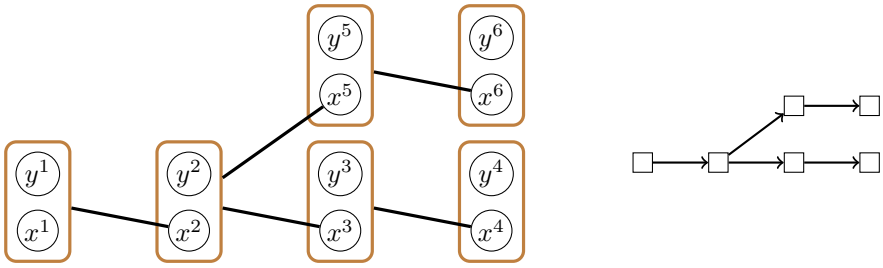


Figure 2.8: An example of a branching wire with its corresponding propagation digraph.

Henceforth, when we say that a contraction is *preceded* by another contraction, it includes the case that the two contractions are in fact the same. By a straightforward induction, we obtain the following from lemma 46 (and lemma 43).

Lemma 47. *In a partial 4-sequence from G, any contraction involving a pair of vertices in a vertical set V has to be preceded by the contraction of the vertical pair of every vertical set V' such that there is a directed path from V' to V in $\mathcal{D}(G)$.*

Contracting wires. As the roots and leaves of wires will be connected to other gadgets, we postpone the description of how to contract wires until after building the overall construction G. Intuitively though, contracting a wire (in the vacuum) consists of contracting the vertical pair of its root, then its fence gadget by applying lemma 45, and finally recursively contracting the subtrees rooted at its children. Since $\mathcal{D}(G)$ has total degree at most 3, every vertex has red degree at most 4 (for the at most 3 adjacent vertical sets, plus the pendant vertex of the fence gadget).

2.3.3 Binary AND gate

The *binary* AND *gate* (AND gadget, for short) simply consists of three vertical sets V^1, V^2, V^3 with an arc from V^1 to V^3, and an arc from V^2 to V^3. As usual, the vertical pairs of V^1, V^2, and V^3, are $\{x^1, y^1\}$, $\{x^2, y^2\}$, and $\{x^3, y^3\}$, respectively. We call the vertical sets V^1, V^2 the *inputs* of the AND gadget, and the vertical set V^3 the *output* of the AND gadget. See fig. 2.9 for an illustration.

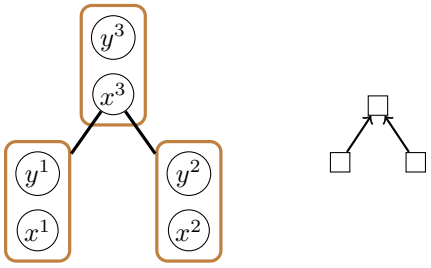


Figure 2.9: An AND gadget, and the corresponding propagation digraph.

Constraint of the AND gadget on a 4-sequence. By lemma 46, we readily derive:

Lemma 48. *Assume G contains an AND gadget with inputs V^1, V^2, and output V^3. In a partial 4-sequence from G, any contraction involving two vertices of V^3 has to be preceded by the contraction of x^1 and y^1, and the contraction of x^2 and y^2.*

Contraction of an AND gadget. Once V^1 and V^2 are contracted into single vertices, one can contract the vertical pair x^3, y^3. This results in a vertex of red degree 2. Thus one can contract the fence gadget of V^3 by applying lemma 45.

2.3.4 Binary OR gate

The *binary* OR *gate* (OR *gadget*) is connected to three vertical sets: two *inputs* V^1, V^2, and one *output* V^3. We start by building two vertical sets V, V' whose vertical pairs are $\{a, b\}$ and $\{c, d\}$, respectively. The edges ac and bd are added, as well as a vertex e adjacent to a and to c. Finally a fence is attached to $\{e\} \cup V \cup V'$.

The OR gadget is connected to its inputs and output, in the following way. Vertex a is made adjacent to x^1 and to y^1 (but not to their fence gadget). Similarly vertex b is linked to x^2 and y^2. Finally x^3 is adjacent to all the vertices of the OR gadget, that is, $\{e\} \cup V \cup V'$ plus the vertices of the outermost fence. See fig. 2.10 for a representation of the OR gadget.

Constraint of the OR gadget on a 4-sequence. By design, one can only start contracting the OR gadget after the vertical pair of at least one of its two inputs has been contracted. This implies that no contraction can involve V^3 before at least one the vertical pairs $\{x^1, y^1\}$ and $\{x^2, y^2\}$ is contracted.

Lemma 49. *Assume G contains an OR gadget attached to inputs V^1 and V^2. In a partial 4-sequence from G, the contractions of a, b and of c, d have to be preceded by the contraction of x^1, y^1 or the contraction of x^2, y^2.*

Proof. Assume none of the pairs $\{a, b\}, \{c, d\}, \{x^1, y^1\}, \{x^2, y^2\}$ have been contracted. Because of the fences, by lemma 44, all the vertices $x^1, y^1, a, b, c, d, x^2, y^2$ and e are in distinct parts. Therefore contracting a and b would create a vertex of red degree at least 5, considering the (singleton) parts of x^1, y^1, c, d, e. Symmetrically contracting c and d yields at least five red neighbors, considering the (singleton) parts of x^2, y^2, a, b, e. □

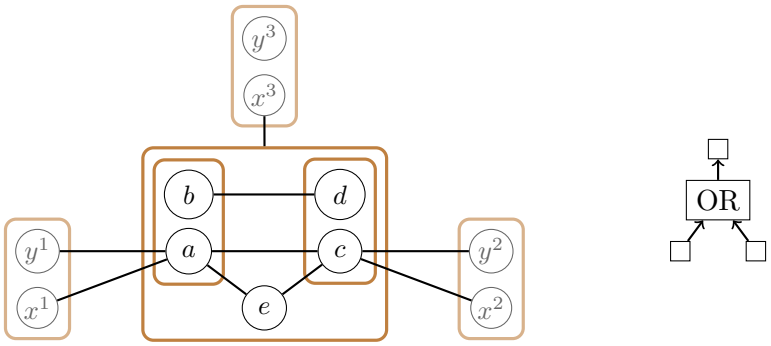


Figure 2.10: An OR gadget attached to inputs V^1, V^2 and output V^3, and its symbolic representation. These vertical sets are technically not part of the OR gate, so we represent them slightly dimmer. The outermost fence of the OR gadget can only be contracted when at least one the pairs x^1, y^1 and x^2, y^2 have been contracted. Only after that, can x^3 and y^3 be contracted together.

From lemma 49 we get the following.

Lemma 50. *Assume G contains an OR gadget attached to inputs V^1 and V^2. In a partial 4-sequence from G, no contraction involving a vertex of V^3 can happen before either the pair x^1, y^1 or the pair x^2, y^2 is contracted.*

Proof. Suppose neither x^1, y^1 nor x^2, y^2 is contracted. By lemma 49, the pairs $\{a, b\}$ and $\{c, d\}$ cannot be contracted. By the first item of lemma 44, no contraction can involve a vertex of $\{a, b, c, d, e\}$. As x^3 is adjacent to all these vertices but not y^3, one cannot contract the vertical pair $\{x^3, y^3\}$. Hence by lemma 43 no contraction can involve a vertex of V^3. □

Contraction of the OR gadget. We now show how to contract the OR gate when the vertical pair of one of its inputs has been contracted.

Lemma 51. *Assume that x^1 and y^1 have been contracted into z^1, and that z^1, x^2, and y^2 all have red degree at most 3. Then there is a partial 4-sequence that contracts the whole OR gadget to a single vertex with only three red neighbors: z^1, x^2, and y^2. (The same holds symmetrically if x^2 and y^2 have been contracted into a single vertex.)*

Proof. First contract a and b into vertex α of red degree 4. At this point the fence of $\{a, b\}$ cannot be contracted yet, as this would make the red degree of α go above 4. Hence we next contract c and d into γ, decreasing the red degree of α to 3. Note that γ has only 4 red neighbors: α, e, x^2, y^2.

By lemma 45, we can now contract the fence gadget of $\{a, b\}$ to a single vertex. Next we contract this latter vertex with α, and the resulting vertex with t; we call α' the obtained vertex. Now γ has only red degree 3, so we can contract the fence gadget of $\{c, d\}$ to a single vertex that we further contract with γ; we call γ' the obtained vertex. We contract α' and γ' in a vertex ε of red degree 3; its three red neighbors are z^1, x^2, and y^2. Again by lemma 45, the outermost fence of the OR gadget can be contracted into a single vertex, that we finally contract with ε. This results in a vertex with three red neighbors: z^1, x^2, and y^2.

Throughout this process the red degree of z^1, x^2, and y^2 never goes above 4. Indeed the red degree of these vertices is initially at most 3, while they have exactly one black neighbor in the entire OR gadget (so at most one part to be in conflict with). □

2.3.5 Variable gadget

We first describe the variable gadget for, say, a variable x. We start by attaching a fence on a set formed by three vertices: $x, \top, \bot$. We add two disjoint copies of an OR gadget, with vertices $a^\top, b^\top, c^\top, d^\top, e^\top$ (resp. $\{a^\bot, b^\bot, c^\bot, d^\bot, e^\bot\}$) plus the vertices in the fence gadgets; see section 2.3.4 and fig. 2.10. We call T', respectively U', the vertex sets of the two OR gadgets. We link $a^\top$ to x and to $\top$, and we link $a^\bot$ to x and to $\bot$. We then add a vertex $f^\top$ (resp. $f^\bot$), make it adjacent to $c^\top$ (resp. $c^\bot$), and add a fence $F'^\top$ (resp. $F'^\bot$) attached to $T' \cup \{f^\top\}$ (resp. $U' \cup \{f^\bot\}$). We add another vertex $g^\top$ (resp. $g^\bot$), make it adjacent to $c^\top$ (resp. $c^\bot$), and add a fence $F^\top$ (resp. $F^\bot$) attached to $T = T' \cup V(F'^\top) \cup \{f^\top, g^\top\}$ (resp. $U = U' \cup V(F'^\bot) \cup \{f^\bot, g^\bot\}$).

The variable gadget is *half-guarded* by two vertical sets V^1, V^2 (with vertical pairs x^1, y^1 and x^2, y^2): We make $\top$ adjacent to x^1 and y^1, and we make $\bot$ adjacent to x^2 and y^2. Finally T guards a vertical set V^3 (with vertical pair x^3, y^3), and U guards a vertical set V^4 (with vertical pair x^4, y^4), that is, x^3 is fully adjacent to T, and x^4 is fully adjacent to U. Vertical set V^3 is the root of what we call the *wire of literal* $+x$, which is the maximal wire containing V^3. We add the $+$ to differentiate the *literal* from the *variable* x. Similarly, V^4 is the root of the *wire of literal* $\neg x$, that is, the maximal wire containing V^4.

This finishes the construction of the variable gadget; see fig. 2.11 for an illustration.

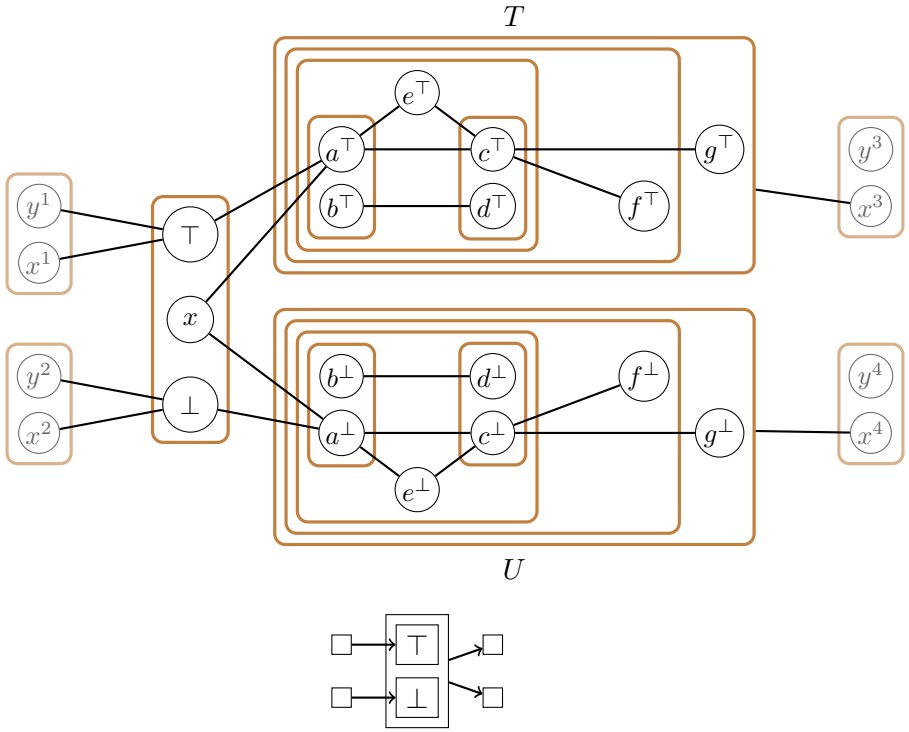


Figure 2.11: Top: A variable gadget half-guarded by V^1, V^2, and with outputs V^3, V^4. Bottom: Its compact representation.

Constraints of the variable gadget on a 4-sequence. Because of the fence gadget attached to $\{\top, x, \bot\}$, one has at some point to contract $\top$ and $\bot$ (be it with or without x). A first observation is that this has to wait that the vertical pair of V^1 or V^2 is contracted.

Lemma 52. *Assume G has a variable gadget half-guarded by vertical sets V^1 and V^2.*

In a partial 4-sequence from G, the contraction of $\top$ and $\bot$ has to be preceded by the contraction of the pair x^1, y^1 or of the pair x^2, y^2.

Proof. The pairs $\{x^1, y^1\}$, $\{x^2, y^2\}$, and $\{\top, \bot\}$ are contained in three disjoint sets S^1, S^2, S, respectively, to which a fence is attached. Thus before any of these pairs are contracted, by lemma 44, a vertex outside $S^1 \cup S^2 \cup S$, like $a^\bot$, is in a different part than the six vertices $x^1, y^1, x^2, y^2, \top, \bot$. Therefore contracting $\top$ and $\bot$ would create a vertex with five red neighbors, considering the parts of $x^1, y^1, x^2, y^2, a^\bot$. □

We next show that no contraction is possible in U (resp. in T), while x and $\bot$ (resp. x and $\top$) are not contracted.

Lemma 53. *In a partial 4-sequence, the contractions of $a^\bot$ and $b^\bot$ (resp. $a^\top$ and $b^\top$) and of $c^\bot$ and $d^\bot$ (resp. $c^\top$ and $d^\top$) have to be preceded by the contraction of x and $\bot$ (resp. x and $\top$). Therefore no contraction is possible in U (resp. T) before x and $\bot$ (resp. x and $\top$) are contracted.*

Proof. Since the two statements are symmetric, we only show it with $\bot$. Assume none of the pairs $\{x, \bot\}$, $\{a^\bot, b^\bot\}$, $\{c^\bot, d^\bot\}$ are contracted. Because of the fence gadgets, by the first item of lemma 44, the vertices $x, \bot, a^\bot, b^\bot, c^\bot, d^\bot, e^\bot, f^\bot$ are pairwise in distinct parts. Therefore contracting $a^\bot$ and $b^\bot$ or $c^\bot$ and $d^\bot$ would create a vertex of red degree at least 5. The structure of the fence gadgets in U thus prevents any contraction. □

We deduce that priming the wire of $\neg x$ (resp. $+x$) can only be done after x and $\bot$ (resp. x and $\top$) are contracted.

Lemma 54. *Assume that G has a variable gadget with outputs the vertical sets V^3 (root of the wire of $+x$) and V_4 (root of the wire of $\neg x$). In a partial 4-sequence from G, the contraction of x^4 and y^4 (resp. x^3 and y^3) has to be preceded by the contraction of x and $\bot$ (resp. x and $\top$).*

Proof. By the second item of lemma 44 applied to the fence attached to U, the pair x^4, y^4 cannot be contracted until U is not contracted to a single vertex. Thus by lemma 53, the pair x^4, y^4 can only be contracted after the pair $x, \bot$ is contracted. The other statement is obtained symmetrically. □

Contraction of the variable gadget. We show two options to contract a "half" of the variable gadget, either T and its fence, or U and its fence, into a single vertex.

Lemma 55. *There is a partial 4-sequence that contracts x and $\top$ together, and $T \cup F^\top$ into a single vertex. Symmetrically there is a partial 4-sequence that contracts x and $\bot$ together, and $U \cup F^\bot$ into a single vertex.*

Proof. We first contract x and $\top$ into a vertex $+x$. Observe that $+x$ has exactly three red neighbors: x^1, y^1, and $a^\bot$. Thus $\{a^\top, b^\top, c^\top, d^\top, e^\top\}$ and their three fences can be contracted exactly like an OR gadget. So by lemma 51, there is a partial 4-sequence that contracts all these vertices to a single vertex u, with three red neighbors ($+x$, $f^\top$, and $g^\top$). We can now contract u and $f^\top$ into u', followed by contracting their fence gadget $F'^\top$ into a single vertex, by lemma 45. That pendant vertex can be contracted to u', and the result to $g^\top$, forming vertex v. Finally, again by lemma 45, the fence $F^\top$ attached to T can be contracted into a single vertex, which can be contracted with v.

The other sequence is the symmetric. □

We now see how the second "half" of the variable gadget can be contracted once the vertical pairs of the *half-guards* V^1, V^2 have been contracted.

Lemma 56. *Assume $T \cup F^\top$ (resp. $U \cup F^\bot$) has been contracted into a single vertex u, and that the pairs $\{\top, x\}$ (resp. $\{\bot, x\}$), $\{x^1, y^1\}$, and $\{x^2, y^2\}$ have been contracted into $+x$ (resp. $\neg x$), z^1, and z^2, respectively. We further assume that the red degree of z^2 (resp. z^1) is at most 3. Then there is partial 4-sequence that contracts $\top, x, \bot$ and their fence into a single vertex, and $U \cup F^\bot$ (resp. $T \cup F^\top$) into a single vertex.*

Proof. We contract $\bot$ with $+x$ into v of red degree 4. This increases the red degree of z^2 by one, which remains at most 4. We then contract $U \cup F^\bot$ into a single vertex w, like in lemma 55. We contract u and w into y, a vertex of red degree at most 3. Now v has degree 3. So we can contract its fence gadget by lemma 45. We further contract v with its pendant neighbor, and finally with y. What results is a unique vertex with four red neighbors.

The other partial sequence is symmetric. □

2.3.6 Clause gadget

A *3-clause gadget* (or simply *clause gadget*) has for *inputs* three vertical sets V^1, V^2, V^3, and for *output* one vertical set V^4. It consists of combining two OR gadgets, and using long chains to make the OR gadgets *distant enough*. We add an OR gadget with input V^1 and V^2, and output V. We then add a long chain from V to V', and an OR gadget with input V' and V^3, and output V^4. The clause gadget is depicted in fig. 2.12. We call *first OR gadget* of the clause gadget, the one with output V, and *second OR gadget*, the one with output V^4.

Figure 2.12: Left: A 3-clause gadget. Right: A shorthand for the gadget.

Constraint of the clause gadget on a 4-sequence. As a consequence of lemmas 47 and 50, we get that once a contraction involves the output, at least one of the vertical pairs of the inputs has been contracted.

Lemma 57. *Assume G contains a clause gadget with inputs V^1, V^2, V^3 and output V^4. In a partial 4-sequence from G, any contraction involving a vertex of V^4 is preceded by the contraction of the vertical pair of one of V^1, V^2, or V^3.*

Proof. Assume a contraction involves a vertex of V^4. By lemma 50 applied to the second OR gadget, the vertical pair of V^3 or of V' has to be contracted beforehand. If the vertical pair V^3 has been contracted, we conclude. Otherwise, the vertical pair of V' has been contracted, and by lemma 47, this implies that the vertical pair of V has been contracted. By lemma 50 applied to the first OR gadget, this in turn implies that the vertical pair of V^1 or of V^2 has been contracted. □

Contraction of the clause gadget. The OR gates of the clause gadgets will be contracted as specified in lemma 51, while we wait the overall construction to describe the contraction of the wires.

2.3.7 Overall construction and correctness

Let $I = (C_1, \dots, C_m)$ be an instance of 3-SAT, that is, a collection of m 3-clauses over the variables $x_1, \dots, x_n$. We further assume that each variable appears at most twice positively, and at most twice negatively in I. The 3-SAT problem remains NP-complete with that restriction, and without $2^{o(n)}$-time algorithm unless the ETH fails; see theorem 30. We build a trigraph G that has twin-width at most 4 if and only if I is satisfiable. As trigraph G will satisfy the condition of lemma 37, it can be replaced by a graph on $O(|V(G)|)$ vertices. We set L the length of the long chain to $2\lceil \log(5n + 3m) \rceil = O(\log n)$.

Construction

We now piece the gadgets described in the previous sections together.

Variable to clause gadgets. For every variable x_i, we add a variable gadget half-guarded by V_i^1, V_i^2, and with outputs V_i^3, V_i^4. We add a long chain starting at vertical set V_i^3 (resp. V_i^4), and ending at a branching vertical set from which starts two long chains stopping at vertical sets $V_i^{\top,1}$ and $V_i^{\top,2}$ (resp. $V_i^{\bot,1}$ and $V_i^{\bot,2}$). Vertical set $V_i^{\top,1}$ (resp. $V_i^{\bot,1}$) serves as the input of the first clause gadget in which x_i appears positively (resp. negatively), while $V_i^{\top,2}$ (resp. $V_i^{\bot,2}$) becomes the input of the second clause gadget in which x_i appears positively (resp. negatively). If a literal has only one occurrence, then we omit the corresponding vertical set, and the long chain leading to it. We nevertheless assume that each literal has at least one occurrence, otherwise the corresponding variable could be safely assigned.

Clause gadgets to global output. For every $j \in [m]$, we add a long chain from the output of the clause gadget of C_j, to a vertical set, denoted by V_j^c. For every $j \in [2, m]$, we then add a long chain starting at V_j^c and ending at V_{j-1}^c. We add an arc from V_1^c to a new vertical set V^o, which we call the *global output*.

Global output back to half-guarding the variable gadgets. For every $i \in [n]$, we add two vertical sets $V_i^{1,r}, V_i^{2,r}$, and puts a long chain starting at $V_i^{1,r}$ and ending at $V_i^{2,r}$. We add a long chain from V^o to $V_1^{1,r}$. We also add a long chain from $V_i^{2,r}$ to $V_{i+1}^{2,r}$, for every $i \in [n-1]$. Finally we add a long chain from $V_i^{a,r}$ to V_i^a for every $a \in \{1, 2\}$ and every $i \in [n]$. Recall that V_i^1 and V_i^2 are half-guarding the variable gadget of x_i.

This finishes the construction of the graph $G = G(I)$. See fig. 2.13 for an illustration.

Correctness

If G has twin-width at most 4, then I is satisfiable. Let us consider the trigraph H obtained after the vertical pair of the global output V^o is contracted (see top-right picture of fig. 2.14). This has to happen in a 4-sequence by the first item of lemma 44 applied to the fence of V^o. By lemma 47, no contraction involving vertices of the vertical sets V_i^1, V_i^2 can have happened (for any $i \in [n]$). This is because there is a directed path in the propagation digraph $\mathcal{D}(G)$ from V^o to V_i^1 and V_i^2.

Thus by lemmas 52 and 54, for every variable x_i ($i \in [n]$), at most one of the wire of $+x_i$ and the wire of $\neg x_i$ has been primed. We can define the corresponding truth assignment $\mathcal{A}$: x_i is set to true if the wire of $\neg x_i$ is *not* primed, and to false if instead the

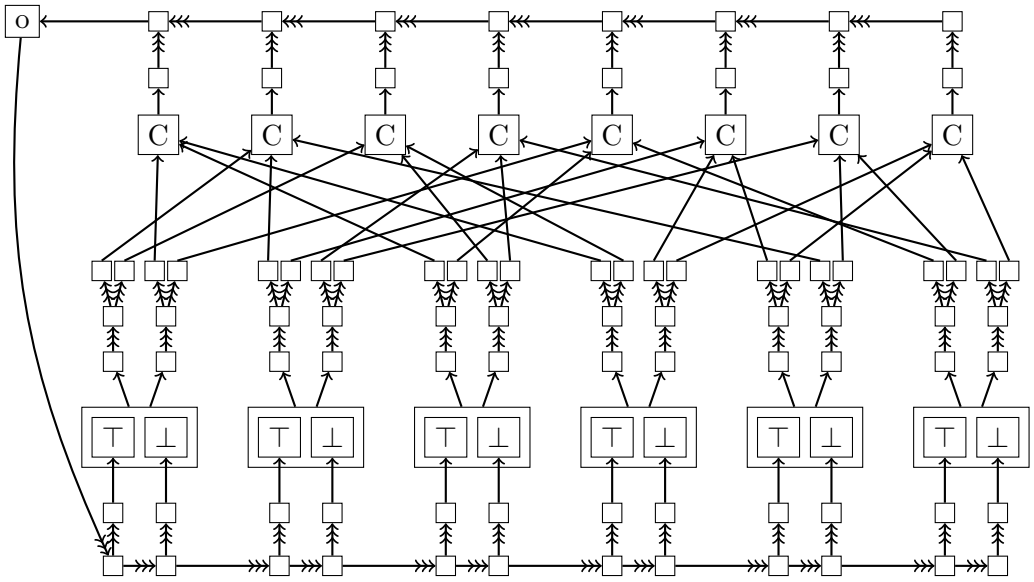


Figure 2.13: An example of the overall construction G on a 3-SAT instance with 6 variables and 8 clauses. The first two clauses are $\neg x_1 \vee x_3 \vee x_4$ and $x_1 \vee x_2 \vee \neg x_5$.

wire of $+x_i$ is *not* primed. Besides, by lemma 47 applied to the contraction in V^o, every vertical pair of a clause-gadget output has been contracted. Then lemma 57 implies that the vertical pair of at least one input of each clause gadget has been contracted. But such a vertical pair can be contracted only if it corresponds to a literal set to true by $\mathcal{A}$. For otherwise, the root of the wire of that literal cannot be contracted. This implies that $\mathcal{A}$ is a satisfying assignment.

If I is satisfiable, then G has twin-width at most 4. In what follows, when we write that we *can* contract a vertex, a set, or make a sequence of contractions, we mean that there is a partial 4-sequence that does the job. Let $\mathcal{A}$ be a satisfying assignment of I. We start by contracting "half" of the variable gadget of each x_i. We add a subscript matching the variable index to the vertex and set labels of fig. 2.11. For every $i \in [n]$, we contract vertices x_i and $\top_i$ together, and $T_i \cup F_i^{\top}$ to a single vertex v_i, if $\mathcal{A}$ sets variable x_i to true, and x_i and $\bot_i$ together, and $U_i \cup F_i^{\bot}$ to a single vertex v_i, if instead $\mathcal{A}$ sets x_i to false. By lemma 55, this can be done by a partial 4-sequence.

Next we contract the wire of $+x_i$ if $\mathcal{A}(x_i)$ is true, or the wire of $\neg x_i$ if $\mathcal{A}(x_i)$ is false. This is done as described in the end of section 2.3.2. We contract the vertical pair of the root of the wire into z_i (the red degree of v_i goes from 1 to 2). We then contract its fence by lemma 45. We can now contract the resulting vertex with z_i. Inductively, we contract the children of the current vertical set to single vertices, in a similar fashion. As the propagation digraph has degree at most 3, this never creates a vertex of red degree more than 4. At this point the trigraph corresponds to the top-left drawing of fig. 2.14.

At the leaves of the wire (the vertical sets $V_i^{\top,1}, V_i^{\top,2}$ or $V_i^{\bot,1}, V_i^{\bot,2}$), we make an exception, and only contract the vertical pair. We then contract, by lemma 51, all the (non-already reduced) OR gadgets (involved in clause gadgets) at least one input of which has its vertical pair contracted. After that, applying lemma 45, we finish the contraction of the OR inputs whose vertical pairs was contracted. Next we contract each output of a contracted OR gadget into a single vertex.

In those clauses where the third literal is not satisfied by $\mathcal{A}$, the output of the clause gadget is not contracted at that point. However, as $\mathcal{A}$ is a satisfying assignment, the

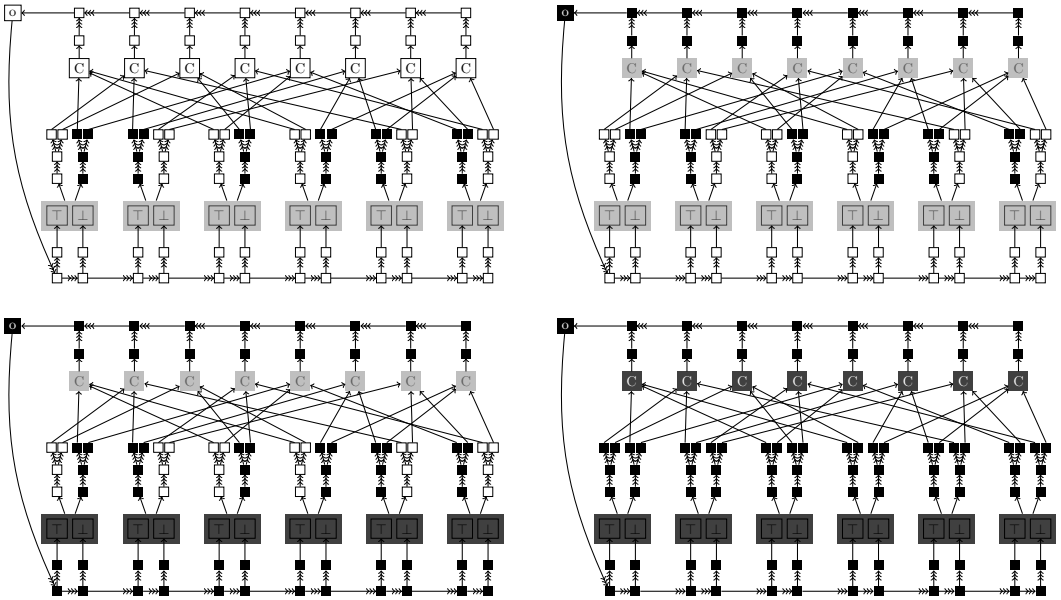


Figure 2.14: The different stages of the contraction sequence. Gadgets in black are contracted to single vertices, while gadgets in gray are only partially contracted.

output of the first OR gate of the gadget is contracted. We then contract each vertical set of the long chain leading to the input of the second OR gate. We only contract the vertical pair of that input, and contract the incident OR gadget, by lemma 51. We finally proceed by contracting the input vertical set into a single vertex, and the output vertical set into a single vertex.

At this point, each output of the clause gadgets is contracted into a single vertex. The (*not* strongly) connected component C of the propagation digraph $\mathcal{D}(G)$ containing the global output V_o is acyclic and has total degree at most 3. All the vertical sets of C with in-degree 0 (ant out-degree 1) in $\mathcal{D}(G)$ are the clause outputs, which have been contracted to single vertices. Thus each vertical set of C can be contracted to single vertices, by repeated use of lemma 45, followed by contracting the pendant vertex (resulting from the fence contraction) with its unique (red) neighbor.

Note that this process terminates by contracting the half-guards V_i^1, V_i^2 (for every $i \in [n]$). The conditions of lemma 56 are now satisfied, so we can finish contracting each variable gadget into two vertices that we further contract together. This results in a vertex of red degree 4. See top-right and bottom-left drawings of fig. 2.14 for the additional contractions leading to that point.

We can then contract the wire of the literal that was set to false by $\mathcal{A}$. Again, we only contract the vertical pair of the inputs of OR gates that are not contracted yet. Then we contract those OR gadgets, and finish by fully contracting the vertical set of those inputs. We next contract each output of the newly contracted OR gates. We eventually contract into a single vertex each vertical set of the long chain in clause gadgets where this was not already done.

At this point, the current trigraph H has only red edges (and corresponds to the bottom-right drawing of fig. 2.14). Thus it can be interpreted as a graph, and we write *degree* instead or *red degree*. H has $4n + 3m$ vertices of degree 3, n vertices of degree 4, and the rest of its vertices have degree 2. Because we added long chains to separate what now corresponds to vertices of degree at least 3, H is an $(\geqslant L)$-subdivision of a graph on $5n+3m$ vertices. Since $L = 2\lceil \log(5n+3m) \rceil$, by lemma 7, H finally admits a 4-sequence.

Wrapping up

The initial trigraph G comprises $O((n+m)L) = O(n \log n)$ gadgets and vertical sets, each consisting of $O(1)$ vertices. Hence $|V(G)| = O(n \log n)$. It is immediate that the construction of G can be made in polynomial time.

The red graph of G is a disjoint union of paths of length 12 (fence gadgets), and isolated vertices (the rest of the trigraph). Thus, by lemma 37, there is a *graph* G' on $O(|V(G)|) = O(n \log n)$ vertices such that G' has twin-width at most 4 if and only if (G has twin-width at most 4 if and only if) I is satisfiable. This concludes the proof of theorem 29 since a $2^{o(N/\log N)}$-time algorithm deciding if an N-vertex graph has twin-width at most 4, would allow to decide 3-SAT where each variable appears at most twice positively and at most twice negatively, in time $2^{o(\frac{n \log n}{\log n + \log\log n})} = 2^{o(n)}$, contradicting the ETH.

Chapter 3

Approximation algorithms parameterized by twin-width

Contents

In this chapter we give the first in-depth study of the approximability of problems in graphs of bounded twin-width. Prior to this work, essentially the only such result was a polynomial-time $O(1)$-approximation algorithm for Min Dominating Set [14].

For any $\varepsilon > 0$, we give a polynomial-time n^{ε}-approximation algorithm for Max Independent Set in graphs of bounded twin-width given with an $O(1)$-sequence. This result is derived from the following time-approximation trade-off: We establish an $O(1)^{2^q-1}$-approximation algorithm running in time $\exp(O_q(n^{2^{-q}}))$, for every integer $q \geqslant 0$. Guided by the same framework, we obtain similar approximation algorithms for Min Coloring and Max Induced Matching. In general graphs, all these problems are known to be highly inapproximable: for any $\varepsilon > 0$, a polynomial-time $n^{1-\varepsilon}$-approximation for any of them would imply that P=NP [58, 59, 60]. We generalize the algorithms for Max Independent Set and Max Induced Matching to the independent (induced) packing of any fixed connected graph H.

In contrast, we show that such approximation guarantees on graphs of bounded twin-width given with an $O(1)$-sequence are very unlikely for Min Independent Dominating Set, and somewhat unlikely for Longest Path and Longest Induced Path. Regarding the existence of better approximation algorithms, there is a (very) light evidence that the obtained approximation factor of n^{ε} for Max Independent Set may be best possible.

3.1 Introduction

Classes of binary structures with bounded twin-width include graph classes with bounded treewidth, and more generally bounded clique-width, proper minor-closed classes, posets with antichains of bounded size, strict subclasses of permutation graphs, as well as $\Omega(\log n)$-subdivisions of n-vertex graphs [4], and some classes of (bounded-degree) expanders [5]. A notable variety of geometrically defined graph classes have bounded twin-width such as map graphs, bounded-degree string graphs [4], classes with bounded queue number or bounded stack number [5], segment graphs with no $K_{t,t}$ subgraph, visibility graphs of 1.5D terrains without large half-graphs, visibility graphs of simple polygons without large independent sets [13].

For every class $\mathcal{C}$ mentioned so far, $O(1)$-sequences can be computed in polynomial time[1] on members of $\mathcal{C}$. For classes of binary structures including a binary relation interpreted as a linear order on the domain (called *ordered binary structures*), there is a fixed-parameter approximation algorithm for twin-width [8]. More precisely, given a graph G and an integer k, there are computable functions f and g such that one can output an $f(k)$-sequence of G or correctly report that $\mathrm{tww}(G) > k$ in time $g(k)n^{O(1)}$. Such an approximation algorithm is currently missing for classes of general (not necessarily ordered) binary structures, and in particular for the class of all graphs.

We will therefore assume that the input graph is given with a d-sequence, and treat d as a constant (or that the input comes from any of the above-mentioned classes). Thus far, this is the adopted setting when designing faster algorithms on bounded twin-width graphs [4, 14, 20, 19, 61]. From the inception of twin-width [4] –actually already from the seminal work of Guillemot and Marx [3]– it was clear that structures wherein this invariant is bounded may *often* allow the design of parameterized algorithms. More concretely, it was shown [4] that, on graphs G given with a d-sequence, model checking a first-order sentence φ is fixed-parameter tractable –it can be solved in time $f(d, \varphi) \cdot n$–, the special cases of, say, k-Independent Set or k-Dominating Set admit single-exponential parameterized algorithms [14], an effective data structure almost linear in n can support constant-time edge queries [20], the triangles of G can be counted in time $O(d^2n + m)$ [19].

So far, however, the connection between *having bounded twin-width* and *enjoying enhanced approximation factors* was tenuous. The only such result concerned Min Dominating Set, known to be inapproximable in polynomial-time within factor $(1 - o(1)) \ln n$ unless P=NP [62], but yet admits a constant-approximation on graphs of bounded twin-width given with an $O(1)$-sequence [14]. We start filling this gap by designing approximation algorithms on graphs of bounded twin-width given with an $O(1)$-sequence for notably Max Independent Set (MIS, for short), Max Induced Matching, and Coloring. Getting better approximation algorithms for MIS and Coloring in that particular scenario was raised as an open problem [14]. Before we describe our results and elaborate on the developed techniques, let us briefly present the notorious inapproximability of these problems in general graphs.

MIS and Coloring are NP-hard [63], and very inapproximable: for every $\varepsilon > 0$, it is NP-hard to approximate these problems within ratio $n^{1-\varepsilon}$ [58, 59]. The same was shown to hold for Max Induced Matching [60]. Besides, there is only little room to improve over the brute-force algorithm in $2^{O(n)}$: Unless the Exponential Time Hypothesis[2] [55] (ETH) fails, no algorithm can solve MIS in time $2^{o(n)}$ [56] (nor the other two problems). For any r (possibly a function of n) WMIS can be r-approximated in time $2^{O(n/r)}$ [64,

[1] Admittedly, for the geometric classes, a representation is (at least partially) needed.

[2] That is, the assumption that there is a $\delta > 0$ such that n-variable 3-SAT cannot be solved in time δ^n.

65]. Bansal et al. [66] essentially shaved a $\log^2 r$ factor to the latter exponent. It is known though that polynomial shavings are unlikely. Charlermsook et al. [67] showed that, for any $\varepsilon > 0$ and sufficiently large r (again r can be function of n), an r-approximation for MIS and Max Induced Matching cannot take time $2^{O(n^{1-\varepsilon}/r^{1+\varepsilon})}$, unless the ETH fails. For instance, investing time $2^{O(\sqrt{n})}$, one cannot hope for significantly better than a $\sqrt{n}$-approximation.

Contributions and techniques Our starting point is a constant-approximation algorithm for MIS running in time $2^{O(\sqrt{n})}$ when presented with an $O(1)$-sequence, which is very unlikely to hold in general graphs by the result of Charlermsook et al. [67].

Theorem 58. *On n-vertex graphs given with a d-sequence Max Independent Set can be $O_d(1)$-approximated in time $2^{O_d(\sqrt{n})}$.*

Our algorithm builds upon the functional equivalence between twin-width and the so-called *versatile twin-width* [5]. We defer the reader to section 3.2 for a formal definition of versatile twin-width. At this point, for our purpose, one only needs to know the following useful consequence of that equivalence. From a d'-sequence of G, we can compute in polynomial time another partition sequence $\mathcal{P}_n, \ldots, \mathcal{P}_1$ of G of width $d := f(d')$, for some computable function f, such that for every integer $1 \leqslant i \leqslant n$, all the i parts of $\mathcal{P}_i$ have size at most $d \cdot \frac{n}{i}$. Even if some parts of P_i can be very small, this partition is balanced in the sense that no part can be larger than d times the part size in a perfectly balanced partition. Of importance to us is $\mathcal{P}_{\lfloor\sqrt{n}\rfloor}$ when the number of parts ($\lfloor\sqrt{n}\rfloor$) and the size of a larger part in the partition (at most $d\frac{n}{\lfloor\sqrt{n}\rfloor} \approx d\sqrt{n}$) are somewhat level.

We can then properly color the red graph (made by the red edges on the vertex set $\mathcal{P}_{\lfloor\sqrt{n}\rfloor}$) with $d+1$ colors. Any color class X is a subset of parts of $\mathcal{P}_{\lfloor\sqrt{n}\rfloor}$ such that between two parts there are either all edges (black edge) or no edge at all (non-edge). In graph-theoretic terms, the subgraph G_X of G induced by all the vertices of all the parts of X have a simple modular decomposition: a partition of at most $\sqrt{n}$ modules each of size at most $d\sqrt{n}$. It is thus routine to compute a largest independent set of G_X essentially in time exponential in the maximum between the number of modules and the maximum size of a module, that is, in at most $d\sqrt{n}$. As one color class X^* contains more than a $\frac{1}{d+1}$ fraction of the optimum, we get our $d+1$-approximation when computing a largest independent set of G_{X^*}. Figure 3.1 serves as a visual summary of what we described so far.

The next step is to substitute recursive calls of our approximation algorithm to exact exponential algorithms on induced subgraphs of size $O_d(\sqrt{n})$. Following this inductive process at depth $q = 2, 3, 4, \ldots$, we degrade the approximation ratio to $(d+1)^3, (d+1)^7, (d+1)^{15}$, etc. but meanwhile we boost the running time to $2^{O_d(n^{1/4})}, 2^{O_d(n^{1/8})}, 2^{O_d(n^{1/16})}$, etc. In effect we show by induction that:

Theorem 59. *On n-vertex graphs given with a d-sequence Max Independent Set has an $O_d(1)^{2^q-1}$-approximation algorithm running in time $2^{O_{d,q}(n^{2^{-q}})}$, for every integer $q \geqslant 0$.*

It is convenient that in the case of MIS we are only making recursive calls on induced subgraphs of the current graph. Hence the twin-width, hence the versatile twin-width bound, can only decrease. And we can simply compute (from scratch) new *balanced* partition sequences for the subinstances. The following polynomial-time algorithm is a corollary of theorem 59 choosing $q = O_{d,\varepsilon}(\log\log n)$.

Theorem 60. *For every $\varepsilon > 0$, Max Independent Set can be n^{ε}-approximated in polynomial-time $O_{d,\varepsilon}(1) \cdot \log^{O_d(1)} n \cdot n^{O(1)}$ on n-vertex graphs given with a d-sequence.*

Note that the exponent of the polynomial factor is an absolute constant (not depending on d nor on ε).

We then apply our framework to Coloring and Max Induced Matching.

Theorem 61. *For every $\varepsilon > 0$, Coloring and Max Induced Matching admit polynomial-time n^{ε}-approximation algorithms on n-vertex graphs of bounded twin-width given with an $O(1)$-sequence.*

The main additional difficulty for Coloring is that one cannot satisfactorily solve/approximate that problem on a modular decomposition by simply coloring its modules and its quotient graph. One needs to tackle a more general problem called Set Coloring. Fortunately this generalization is the fixed point we are looking for: approximating Set Coloring can be done in our framework by mere recursive calls (to itself).

For Max Induced Matching, we face a new kind of obstacle. It can be the case that no decent solution is contained in any color class X –in the chosen $d+1$-coloring of the red graph $G/\mathcal{P}_{\lfloor\sqrt{n}\rfloor}$. For instance, it is possible that any such color class X induces in G an edgeless graph, while very large induced matchings exist with endpoints in two distinct color classes. We thus need to also find large induced matchings within the black edges and within the red edges of $G/\mathcal{P}_{\lfloor\sqrt{n}\rfloor}$. This leads to a more intricate strategy intertwining the coloring of bounded-degree graphs (specifically the red graph and the square of its line graph) and recursive calls to induced subgraphs of G, and to special induced subgraphs of the total graph (i.e., made by both the red and black edges) of $G/\mathcal{P}_{\lfloor\sqrt{n}\rfloor}$. Although this is not necessary, one can observe that the latter graphs are also induced subgraphs of G itself.

We then explore the limits of our results and framework in terms of amenable problems. We give the following technical generalization to the approximation algorithms for MIS and Max Induced Matching.

Theorem 62. *For every connected graph H and $\varepsilon > 0$, Mutually Induced H-packing admits a polynomial-time n^{ε}-approximation algorithms on n-vertex graphs of bounded twin-width given with an $O(1)$-sequence.*

In this problem, one seeks for a largest induced subgraph that consists of a disjoint union of copies of H. All the previous technical issues are here combined. We try all the possibilities of batching the vertices of H into at most $|V(H)|$ parts of $G/\mathcal{P}_{\lfloor\sqrt{n}\rfloor}$, based on the trigraph that these parts define. For instance with $H = K_2$ (an edge), i.e., the case of Max Induced Matching, the three possible trigraphs are the 1-vertex trigraph, two vertices linked by a red edge, and two vertices linked by a black edge. In the general case, the problem generalization is quite delicate to find. We have to keep some partitions of $V(G)$ and $V(H)$ to enforce that the copies of H in G follow a pattern that the algorithm committed to higher up in the recursion tree, and a weight function on $|V(H)|$-tuples of vertices of G, not to forget how many mutually induced copies of H can be packed *within* these vertices. The other novelty is that some recursive calls are on induced subgraphs of the total graph of $G/\mathcal{P}_{\lfloor\sqrt{n}\rfloor}$ that are *not* induced subgraphs of G. Fortunately, these graphs keep the same bound of versatile twin-width, and thus our framework allows it.

Defining, for a family of graphs $\mathcal{H}$, Mutually Induced $\mathcal{H}$-packing as the same problem where the connected components of the induced subgraph should all be in $\mathcal{H}$, we get a similar approximation factor when $\mathcal{H}$ is a finite set of connected graphs. (Note that Mutually Induced H-packing is sometimes called Independent Induced H-Packing.) In particular, we can similarly approximate Independent H-Packing, which

is the same problem but the copies of H need not be induced. (Our approximation algorithms could extend to other H-packing variants without the independence requirement, but these problems can straightforwardly be $O(1)$-approximated in general graphs.)

We can handle some cases when $\mathcal{H}$ is infinite, too. For instance, by slightly adapting the case of MIS, we can get an n^ε-approximation when $\mathcal{H}$ is the set of all cliques. We show this more involved example, also expressible as Mutually Induced $\mathcal{H}$-packing for $\mathcal{H}$ the set of all trees or the set all stars.

Theorem 63. *For every $\varepsilon > 0$, finding the induced (star) forest with the most edges admits a polynomial-time n^ε-approximation algorithms on n-vertex graphs of bounded twin-width given with an $O(1)$-sequence.*

As we already mentioned, our framework is exclusively useful for problems that are very inapproximable in general graphs; at least for which an n^ε-approximation algorithm is not known for every $\varepsilon > 0$. Are there natural such problems that cannot be approximated better in graphs of bounded twin-width? We answer this question positively with the example of Min Independent Dominating Set.

Theorem 64. *For every $\varepsilon > 0$, Min Independent Dominating Set does not admit an $n^{1-\varepsilon}$-approximation algorithm in n-vertex graphs given with an $O(1)$-sequence, unless P=NP.*

The reduction is the same as the one for general graphs [68], but performed from a planar variant of 3-SAT. The obtained instances are not planar but can be contracted to planar trigraphs, hence overall have bounded twin-width.

Finally the case of Longest Path and Longest Induced Path is interesting. The best approximation factor for the former [69] is worse than $n^{0.99}$, while the latter is known to have the same inapproximability as MIS [70]. However an n^ε-approximation algorithm (for every $\varepsilon > 0$) is not excluded for Longest Path. We show that the property of bounded twin-width is unlikely to help for these two problems, as it would lead to better approximation algorithms for Longest Path in general graphs. This is mainly because subdividing at least $2 \log n$ times every edge of any n-vertex graph gives a graph with twin-width at most 4 [28].

Theorem 65. *For any $r = \omega(1)$, an r-approximation for Longest Induced Path or Longest Path on graphs given with an $O(1)$-sequence would imply a $(1 + o(1))r$-approximation for Longest Path in general graphs.*

In turn, this can be used to exhibit a family $\mathcal{H}$ with an infinite antichain for the *induced subgraph* relation such that Mutually Induced $\mathcal{H}$-packing is *hard* to n^ε-approximate on graphs of bounded twin-width. The family $\mathcal{H}$ is simply the set of all paths terminated by triangles at both ends.

Theorem 66. *There is an infinite family $\mathcal{H}$ of connected graphs such that if for every $\varepsilon > 0$, Mutually Induced $\mathcal{H}$-packing admits an n^ε-approximation algorithm on n-vertex graphs given with an $O(1)$-sequence, then so does Longest Path on general graphs.*

table 3.1 summarizes our results and hints at future work.

For the main highly inapproximable graph problems, we either obtain an n^ε-approximation algorithm on graphs of bounded twin-width given with an $O(1)$-sequence, or a conditional obstruction to such an algorithm. In the former case, can we improve further the approximation factor? The next theorem was observed using the self-improvement reduction

Problem name	lower bound general graphs	upper bound bounded tww	lower bound bounded tww
Max Independent Set	$n^{1-\varepsilon}$	$\boxed{n^{\varepsilon}}$	?, self-improvement
Coloring	$n^{1-\varepsilon}$	$\boxed{n^{\varepsilon}}$	$4/3-\varepsilon$
Max Induced Matching	$n^{1-\varepsilon}$	$\boxed{n^{\varepsilon}}$	?
Mut. Ind. H-Packing	$n^{1-\varepsilon}$	$\boxed{n^{\varepsilon}\ (H \text{ connected})}$	?
Mut. Ind. $\mathcal{H}$-Packing	$n^{1-\varepsilon}$	$\boxed{n^{\varepsilon} \text{ for some } \mathcal{H}}$	$\boxed{\text{Longest Path-hard}}$
Min Ind. Dom. Set	$n^{1-\varepsilon}$	$n/\mathsf{polylog}(n)$	$\boxed{n^{1-\varepsilon}}$
Longest Path	$2^{\log^{1-\varepsilon} n}$	$n/\exp(\Omega(\sqrt{\log n}))$	$\boxed{\text{Longest Path-hard}}$
Longest Induced Path	$n^{1-\varepsilon}$	$n/\mathsf{polylog}(n)$	$\boxed{\text{Longest Path-hard}}$
Min Dominating Set	$(1-\varepsilon)\ln n$	$O(1)$	?

Table 3.1: Approximability status of graph problems in general graphs and in graphs of bounded twin-width given with an $O(1)$-sequence. Everywhere "ε" should be read as "$\forall \varepsilon > 0$". Our results are enclosed by boxes. "Longest Path-hard" means that getting an r-approximation would yield essentially the same ratio for Longest Path in general graphs. The other lower bounds are under standard complexity-theoretic assumptions, mostly P≠NP. Not to clutter the table, we do not put the references, which can all be found in the paper.

of Feige et al. [71], which preserves the twin-width bound. This reduction consists of going from a graph G to the lexicographic product $G[G]$, where every vertex of G is replaced by a module inducing a copy of G (and iterating this trick).

Theorem 67 ([14])**.** *Let $r : \mathbb{N} \to \mathbb{R}$ be any non-decreasing function such that for every $\varepsilon > 0$, $r(n) = o(n^{\varepsilon})$. If Max Independent Set admits an $r(n)$-approximation algorithm on n-vertex graphs of bounded twin-width given with an $O(1)$-sequence, then it further admits an $r(n)^{\varepsilon}$-approximation.*

To our knowledge, the application of the self-improvement trick is always to strengthen a lower bound, and never to effortlessly obtain a better approximation factor. Therefore, we may take theorem 67 as a weak indication that our approximation ratio is best possible. Still, not even a polynomial-time approximation scheme (PTAS) is ruled out for MIS (nor for Max Induced Matching, Min Dominating Set, etc.) and we would like to see better approximation algorithms. For Coloring, as was previously observed [14], a PTAS is ruled out by the NP-hardness of deciding if a planar graph is 3-colorable or 4-chromatic, since planar graphs have twin-width at most 9 and a 9-sequence can be found in linear time [12].

3.2 Preliminaries

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 $[1, i]$.

3.2.1 Handled graph problems

We will consider several problems throughout the paper. We recall here the definition of the most central ones. Some technical problem generalizations will be defined along the

way.

Weighted Max Independent Set (WMIS, for short)
Input: A graph G and a weight function $V(G) \to \mathbb{Q}$.
Output: A set $S \subseteq V(G)$ such that $\forall u, v \in S$, $uv \notin E(G)$ maximizing $w(S) := \sum_{v \in S} w(v)$.

A feasible solution to WMIS is called an *independent set*. The Max Independent Set (MIS, for short) problem is the particular case with $w(v) = 1$, $\forall v \in V(G)$. We may denote by $\alpha(G)$, the *independence number*, that is the optimum value of WMIS on graph G.

Coloring
Input: A graph G.
Output: A partition $\mathcal{P}$ of $V(G)$ into independent sets minimizing the cardinality of $\mathcal{P}$.

Equivalently, Coloring can be expressed as finding an integer k and a map $c : V(G) \to [k]$ such that for every $uv \in E(G)$, $c(u) \neq c(v)$, while minimizing k.

Max Induced Matching
Input: A graph G, possibly together with a weight function $w : E(G) \to \mathbb{Q}$.
Output: A set $S \subseteq E(G)$ such that $\forall uv \neq u'v' \in S$, $\{u, v\} \cap \{u', v'\} = \emptyset$ and $G[\{u, v, u', v'\}]$ has exactly two edges, maximizing $w(S) := \sum_{e \in S} w(e)$.

An *induced matching* is a pairwise disjoint set of edges (i.e., a matching) with no edge bridging them. We now give a common generalization of WMIS and Max Induced Matching.

Mutually Induced $\mathcal{H}$-packing
Input: A graph G, possibly together with a weight function $w : V(G) \to \mathbb{Q}$.
Output: A set $S \subseteq V(G)$ such that $G[S]$ is a disjoint union of graphs each isomorphic to a graph in $\mathcal{H}$, maximizing $w(S) := \sum_{v \in S} w(v)$.

When $\mathcal{H}$ consists of a single graph, say H, we simply denote the former problem Mutually Induced H-packing. WMIS and Max Induced Matching are the special cases when H is a vertex and an edge, respectively.

3.2.2 Balanced partition sequences

The notion of *versatile twin-width* is a crucial opening step to our algorithms; see [5]. Let us call *d-contraction* a contraction between two trigraphs of maximum red degree at most d. A *tree of d-contractions* of a trigraph G (of maximum red degree at most d) is a rooted tree, whose root is labeled by G, whose leaves are all labeled by 1-vertex trigraphs K_1, and such that one can go from any parent to any of its children by performing a single d-contraction. Observe that d-sequences coincide with trees of d-contractions that are paths. A trigraph G has *versatile twin-width d* if G admits a tree of d-contractions in which every internal node, labeled by, say, F, has at least $|V(F)|/d$ children each obtained by contracting one of a list of $|V(F)|/d$ pairwise disjoint pairs of vertices of F.

It was shown that twin-width and versatile twin-width are functionally equivalent [5]. The relevant consequence for our purposes is that every graph G with a d'-sequence admits a *balanced* d-sequence, where $d = h(d')$ depends only on d', i.e., one for which the partitions $\mathcal{P}_n, \ldots, \mathcal{P}_1$ are such that for every $i \in [n]$ and $P \in \mathcal{P}_i$, $|P| \leqslant d \cdot \frac{n}{i}$. As we will resort to recursion on induced subtrigraphs and quotient trigraphs, we need to keep more

information on those subinstances that the mere fact that they have twin-width at most d (otherwise the twin-width bound could quickly diverge).

This will be done by opening up the proof in [5], and handling divided $0,1,r$-matrices with some specific properties. Thus we need to recall the relevant definitions.

Given two partitions $\mathcal{P}, \mathcal{P}'$ of the same set, we say that $\mathcal{P}'$ is a *coarsening* of $\mathcal{P}$ if every part of $\mathcal{P}$ is contained in a part of $\mathcal{P}'$, and $\mathcal{P}, \mathcal{P}'$ are distinct. Given a matrix M, we call *row division* (resp. *column division*) a partition of the rows (resp. columns) of M into parts of consecutive rows (resp. columns). A (k,ℓ)*-division*, or simply *division*, of a matrix M is a pair $(\mathcal{R} = \{R_1, \dots, R_k\}, \mathcal{C} = \{C_1, \dots, C_\ell\})$ where $\mathcal{R}$ is a row division and $\mathcal{C}$ is a column division. In a matrix division $(\mathcal{R}, \mathcal{C})$, each part $R \in \mathcal{R}$ is called a *row part*, and each part $C \in \mathcal{C}$ is called a *column part*. Given a subset R of rows and a subset C of columns in a matrix M, the *zone* $M[R, C]$ denotes the submatrix of all entries of M at the intersection between a row of R and a column of C. A *zone* of a matrix partitioned by $(\mathcal{R}, \mathcal{C}) = (\{R_1, \dots, R_k\}, \{C_1, \dots, C_\ell\})$ is any $M[R_i, C_j]$ for $i \in [k]$ and $j \in [\ell]$. A zone is *constant* if all its entries are identical, *horizontal* if all its columns are equal, and *vertical* if all its rows are equal. A *0,1-corner* is a 2×2 $0,1$-matrix which is neither horizontal nor vertical.

Unsurprisingly, $0,1,r$*-matrices* are such that each entry is in $\{0,1,r\}$ where r is an error symbol that should be understood as a red edge. A *neat division* of a $0,1,r$-matrix is a division for which every zone either contains only r entries or contains no r entry and is horizontal or vertical (or both, i.e., constant). Zones filled with r entries are called *mixed*. A *neatly divided matrix* is a pair $(M, (\mathcal{R}, \mathcal{C}))$ where M is a $0,1,r$-matrix and $(\mathcal{R}, \mathcal{C})$ is a neat division of M. A t*-mixed minor* in a neatly divided matrix is a (t,t)-division which coarsens the neat subdivision, and contains in each of its t^2 zones at least one mixed zone (i.e., filled with r entries) or a 0,1-corner. A neatly divided matrix is said t*-mixed free* if it does not admit a t-mixed minor.

A *mixed cut of a row part* $R \in \mathcal{R}$ of a neatly divided matrix $(M, (\mathcal{R}, \mathcal{C} = \{C_1, C_2, \dots\}))$ is an index i such that both $M[R, C_i]$ and $M[R, C_{i+1}]$ are not mixed, and there is a $0,1$-corner in the 2-by-$|R|$ zone defined by the last column of C_i, the first column of C_{i+1}, and R. The *mixed value of a row part* $R \in \mathcal{R}$ of a neatly divided matrix $(M, (\mathcal{R}, \mathcal{C} = \{C_1, C_2, \dots\}))$ is the number of mixed zones $M[R, C_j]$ plus the number of mixed cuts between two (adjacent non-mixed) zones $M[R, C_j]$ and $M[R, C_{j+1}]$. We similarly define the mixed value of a column part $C \in \mathcal{C}$. The *mixed value of a neat division* of a $0,1,r$-matrix is the maximum of the mixed values taken over every part. The *part size* of a division $(\mathcal{R}, \mathcal{C})$ is defined as $\max(\max_{R \in \mathcal{R}} |R|, \max_{C \in \mathcal{C}} |C|)$. A division is *symmetric* if the largest row index of each row part and the largest column index of each column part define the same set of integers. We call *symmetric fusion* of a symmetric division the fusion of two consecutive parts in $\mathcal{C}$ and of the two corresponding parts in $\mathcal{R}$. A symmetric fusion on a symmetric division yields another symmetric division. A matrix $A := (a_{i,j})_{i,j}$ is said *symmetric* in the usual sense, namely, for every entry $a_{i,j}$ of A, $a_{i,j} = a_{j,i}$.

In what follows, we set $c_d := 8/3(t+1)^2 2^{4t}$. The following definition is key.

Definition 68. *Let $\mathcal{M}_{n,d}$ be the class of the neatly divided $n \times n$ symmetric $0,1,r$-matrices $(M, (\mathcal{R}, \mathcal{C}))$, such that $(\mathcal{R}, \mathcal{C})$ is symmetric and has:*

- *mixed value at most $4c_d$,*
- *part size at most 2^{4c_d+2}, and*
- *no d-mixed minor.*

The *red number* of a matrix is the maximum number of r entries in a single column or row of the matrix.

Lemma 69. *Let $(M,(\mathcal{R},\mathcal{C})) \in \mathcal{M}_{n,d}$. The red number of M is at most $c_d \cdot 2^{4c_d+4}$. Thus, the trigraph whose adjacency matrix is M has maximum red degree at most $c_d \cdot 2^{4c_d+4}$.*

Proof. Any row or column intersects at most $4c_d$ mixed zones (filled with r entries). Each mixed zone has width and length bounded by the part size 2^{4c_d+2}. Hence the maximum total number of r entries on a single row or column is at most $4c_d \cdot 2^{4c_d+2} = c_d \cdot 2^{4c_d+4}$. □

A *coarsening* of a neatly divided matrix $(M,(\mathcal{R},\mathcal{C}))$ is a neatly divided matrix $(M',(\mathcal{R}',\mathcal{C}'))$ such that $(\mathcal{R}',\mathcal{C}')$ is a coarsening of $(\mathcal{R},\mathcal{C})$, and M' is obtained from M by setting to r all entries that lie, in M divided by $(\mathcal{R}',\mathcal{C}')$, in a zone with at least one r entry or a 0,1-corner. We also refer to the process of going from $(M,(\mathcal{R},\mathcal{C}))$ to $(M',(\mathcal{R}',\mathcal{C}'))$ as *coarsening operation*. A coarsening operation from $(M,(\mathcal{R},\mathcal{C})) \in \mathcal{M}_{n,d}$ to $(M',(\mathcal{R}',\mathcal{C}'))$ is said *invariant-preserving* if $(M',(\mathcal{R}',\mathcal{C}')) \in \mathcal{M}_{n,d}$.

The following lemma is the crucial building block of the current section.

Lemma 70 ([72, Lemma 18])**.** *We set $s := 2^{4c_d+4}$. Every neatly divided matrix $(M,(\mathcal{R},\mathcal{C})) \in \mathcal{M}_{n,d}$ has an invariant-preserving coarsening $(M',(\mathcal{R}',\mathcal{C}')) \in \mathcal{M}_{n,d}$ with $\lfloor n/s \rfloor$ disjoint pairs of identical columns. Given $(M,(\mathcal{R},\mathcal{C}))$, both $(M',(\mathcal{R}',\mathcal{C}'))$ and the pairs of columns can be computed in $n^{O(1)}$ time.*

In [72], it is not explicitly stated that the invariant-preserving coarsening (hence the pairs of identical columns) can be found in polynomial time. However it is easy to check that the proof is effective, since it greedily symmetrically fuses two consecutive parts, provided the resulting divided matrix remains in $\mathcal{M}_{n,d}$. A special case of the following observation is shown in [72, Lemma 19].

Lemma 71. *Let $(M,(\mathcal{R},\mathcal{C})) \in \mathcal{M}_{n,d}$ be a neatly divided matrix. Removing a set of h columns and the h corresponding rows, and possibly removing from the division the parts that are now empty, results in a neatly divided matrix in $\mathcal{M}_{n-h,d}$.*

Proof. By construction, the new matrix and division are symmetric. The new neatly divided matrix remains d-mixed free. The part size and the mixed value can only decrease. □

Lemma 72 ([72, Beginning of Lemma 20])**.** *Given any graph G with a d-sequence, one can find in polynomial-time an adjacency matrix M of G, such that $(M,(\mathcal{R},\mathcal{C}))$ is a neatly divided matrix of $\mathcal{M}_{n,2d+2}$ with $(\mathcal{R},\mathcal{C})$ the finest division of M (i.e., the one where all parts are of size 1).*

The adjacency matrix of a trigraph extends the one of a graph by putting r symbols when the vertices of the corresponding row and column are linked by a red edge. A neatly divided matrix $(M,(\mathcal{R},\mathcal{C}))$ is said *conform* to a trigraph G if M is the adjacency matrix of a trigraph G' such that G is a cleanup of G'. Furthermore, we assume (and keep implicit) that we know the one-to-one correspondence between each row (and corresponding column) of M and vertex of G.

Lemma 73. *Let d be a natural, $s := 2^{4c_d+4}$, and $d' := c_d \cdot 2^{4c_d+4}$. Let G be an n-vertex trigraph given with a neatly divided matrix $(M,(\mathcal{R},\mathcal{C})) \in \mathcal{M}_{n,d}$ conform to G. A partial d'-sequence $\mathcal{S}$ from G to a trigraph H satisfying*

- *$|V(H)| = \lfloor \sqrt{n} \rfloor$, and*
- *$\forall u \in V(H), |u(G)| \leqslant s\sqrt{n}$,*

and a neatly divided matrix $(M',(\mathcal{R}',\mathcal{C}')) \in \mathcal{M}_{\lfloor \sqrt{n} \rfloor,d}$ conform to H can be computed in time $n^{O(1)}$.

Proof. This is a consequence of lemmas 70 and 71; see the proof of the more general lemma 75. □

Combining lemmas 72 and 73, one obtains the following.

Lemma 74. *Let d be a natural, $s := 2^{4c_d+4}$, and $d' := c_d \cdot 2^{4c_d+4}$. Given an n-vertex graph G with a d-sequence, one can compute in time $n^{O(1)}$ a partition $\mathcal{P} = \{P_1, P_2, \ldots, P_{\lfloor\sqrt{n}\rfloor}\}$ of $V(G)$ satisfying*

- *for every integer $1 \leqslant i \leqslant \lfloor\sqrt{n}\rfloor$, $|P_i| \leqslant s\sqrt{n} \leqslant d'\sqrt{n}$, and*
- *the red graph of $G/\mathcal{P}$ has maximum degree at most d'.*

We will need a stronger inductive form of lemma 74, also a consequence of lemmas 72 and 73.

Lemma 75. *Let $\hat{d}$ be a natural, $d = 2\hat{d} + 2$, and set $s := 2^{4c_d+4}$, and $d' := c_d \cdot 2^{4c_d+4}$. Given an n-vertex graph G given with a $\hat{d}$-sequence, or an n-vertex trigraph G with a neatly divided matrix $(M, (\mathcal{R}, \mathcal{C})) \in \mathcal{M}_{n,d}$ such that M is conform to G, one can compute in time $n^{O(1)}$ a partition $\mathcal{P} = \{P_1, P_2, \ldots, P_{\lfloor\sqrt{n}\rfloor}\}$ of $V(G)$ with maximum red degree at most d' satisfying that, for every integer $1 \leqslant i \leqslant \lfloor\sqrt{n}\rfloor$, $|P_i| \leqslant s\sqrt{n} \leqslant d'\sqrt{n}$, and for any trigraph H that is*

- *a cleanup of an induced subtrigraph of $G/\mathcal{P}$, or*
- *an induced subtrigraph $G[\bigcup_{i \in J \subseteq [\lfloor\sqrt{n}\rfloor]} P_i]$,*

a neatly divided matrix $(M', (\mathcal{R}', \mathcal{C}')) \in \mathcal{M}_{|V(H)|,d}$ conform to H can be computed in time $n^{O(1)}$.

Proof. If we are given a graph G with a $\hat{d}$-sequence, we immediately compute a neatly divided matrix $(M, (\mathcal{R}, \mathcal{C})) \in \mathcal{M}_{n,d}$ conform to G, by lemma 72. We then proceed as if we received the second kind of input.

We will build iteratively the partition $\mathcal{P} = \{P_1, P_2, \ldots, P_{\lfloor\sqrt{n}\rfloor}\}$ starting from the finest partition. At each step we merge two parts, until the number of parts is $\lfloor\sqrt{n}\rfloor$. At this point, we have the desired partition $\mathcal{P}$.

We iteratively maintain a trigraph G^z and a neatly divided matrix $(M^z, (\mathcal{R}^z, \mathcal{C}^z)) \in \mathcal{M}_{n-z+1,d}$ conform to it. The maintained partition is just the one corresponding to the parts of G^z. Initially, G^1 is G, and $(M^1, (\mathcal{R}^1, \mathcal{C}^1)) = (M, (\mathcal{R}, \mathcal{C})) \in \mathcal{M}_{n,d}$. At step z we do the following. We apply lemma 70 on $(M^z, (\mathcal{R}^z, \mathcal{C}^z)) \in \mathcal{M}_{n-z+1,d}$ and obtain, in polynomial-time, an invariant-preserving coarsening $(M'^z, (\mathcal{R}'^z, \mathcal{C}'^z)) \in \mathcal{M}_{n-z+1,d}$, and $h := \lfloor (n-z+1)/s \rfloor$ disjoint pairs of equal columns $\{c_1, c'_1\}, \ldots, \{c_h, c'_h\}$ in $(M'^z, (\mathcal{R}'^z, \mathcal{C}'^z))$. Let $\{r_1, r'_1\}, \ldots, \{r_h, r'_h\}$ be the corresponding rows, and $\{v_1, v'_1\}, \ldots, \{v_h, v'_h\}$ the corresponding vertices. Observe that a coarsening of a neatly divided matrix conform to a trigraph is still conform to that trigraph, since the new matrix may only have some r entries in place of some previously 0 or 1 entries. In particular, $(M'^z, (\mathcal{R}'^z, \mathcal{C}'^z))$ is conform to G^z.

There is at least one pair $\{v_i, v'_i\}$ whose contraction forms a part of size at most n/h. Indeed, otherwise the union of the parts corresponding to $v_1, v'_1 \ldots, v_h, v'_h$ is larger than n. We remove c'_i and r'_i from $(M'^z, (\mathcal{R}'^z, \mathcal{C}'^z))$. By lemma 71, we obtain a neatly divided matrix of $M_{n-z,d}$ that we denote by $(M^{z+1}, (\mathcal{R}^{z+1}, \mathcal{C}^{z+1}))$. As we stop when $n - z + 1 = \lfloor\sqrt{n}\rfloor$, it means that the maximum size of a part of our partition is at most $n/h \leqslant sn/\sqrt{n} = s\sqrt{n}$. The bound on the maximum red degree of the obtained partition (actually of all maintained partitions) is given by lemma 69.

We now show to find, for any cleanup H of an induced subtrigraph of $G/\mathcal{P}$, a neatly divided matrix $(M', (\mathcal{R}', \mathcal{C}')) \in \mathcal{M}_{|V(H)|,d}$ conform to H. We first observe, as a consequence

of lemmas 70 and 71, that $(M^{\lfloor\sqrt{n}\rfloor}, (\mathcal{R}^{\lfloor\sqrt{n}\rfloor}, \mathcal{C}^{\lfloor\sqrt{n}\rfloor})) \in \mathcal{M}_{\lfloor\sqrt{n}\rfloor,d}$ is conform to $G/\mathcal{P}$. Taking an induced subgraph H' of $G/\mathcal{P}$ (i.e., removing vertices from it), we get, by removing the corresponding rows and columns in $(M^{\lfloor\sqrt{n}\rfloor}, (\mathcal{R}^{\lfloor\sqrt{n}\rfloor}, \mathcal{C}^{\lfloor\sqrt{n}\rfloor}))$ a neatly divided matrix $(M', (\mathcal{R}', \mathcal{C}')) \in \mathcal{M}_{|V(H')|,d}$ conform to H', by lemma 71. Note finally that taking a cleanup H of H', we can simply keep $(M', (\mathcal{R}', \mathcal{C}'))$ as a neatly divided matrix of $\mathcal{M}_{|V(H)|,d}$ conform to G. The second item, concerning induced subtrigraphs $G[\bigcup_{i\in J\subseteq[\lfloor\sqrt{n}\rfloor]} P_i]$ is a simple application of lemma 71, and works more generally for any induced subtrigraph of G. □

In effect, we will only apply lemma 75 for graphs G and H, i.e., when H is an induced subgraph of G or a full cleanup of an induced subtrigraph of $G/\mathcal{P}$. Indeed, the structures H will correspond to subinstances. We want those to be graphs, so that the tackled graph problem is well-defined on them.

3.2.3 Subexponential-time constant-approximation algorithm

We present a subexponential-time $O_d(1)$-approximation for WMIS on graphs given with a d-sequence, which we recall, is unlikely to exist in general graphs [67].

Lemma 76. *Let d' be a natural, $s := 2^{4c_{d'}+4}$, and $d := c_{d'} \cdot 2^{4c_{d'}+4}$. Assume n-vertex inputs G, vertex-weighted by w, are given with a d'-sequence.* Weighted Max Independent Set *can be $(d+1)$-approximated in time $2^{O_d(\sqrt{n})}$ on these inputs.*

Proof. By lemma 74, we compute in polynomial time a partition $\mathcal{P} = \{P_1, \dots, P_{\lfloor\sqrt{n}\rfloor}\}$ of $V(G)$ whose parts have size at most $s\sqrt{n}$ and such that $\mathcal{R}(G/\mathcal{P})$ has maximum degree at most d.

For every integer $1 \leqslant i \leqslant \lfloor\sqrt{n}\rfloor$, we compute a heaviest independent set in $G[P_i]$, say S_i. Even with an exhaustive algorithm, this takes time $\sqrt{n} \cdot s^2\sqrt{n} \cdot 2^{s\sqrt{n}} = 2^{O_d(\sqrt{n})}$. We then $(d+1)$-color (in linear time) $\mathcal{R}(G/\mathcal{P})$, which is possible since this graph has maximum degree at most d. This defines a coarsening of $\mathcal{P}$ in $d+1$ parts $\mathcal{Q} = \{C_1, \dots, C_{d+1}\}$. Thus, $\mathcal{Q}$ is a partition of $V(G)$ such that C_j consists of all the parts $P_i \in \mathcal{P}$ receiving color j in the $(d+1)$-coloring of $\mathcal{R}(G/\mathcal{P})$.

For every $j \in [d+1]$, let H_j be the *graph* $(G/\mathcal{P})[C_j]$[3] vertex-weighted by $P_i \subseteq C_j \mapsto w(S_i)$. Note that $(G/\mathcal{P})[C_j]$ can indeed be assimilated to a graph, since it has, by design, no red edge. We compute a heaviest independent set in H_j, say R_j. This takes time $(d+1) \cdot n \cdot 2^{\sqrt{n}} = 2^{O_d(\sqrt{n})}$. We output $\bigcup_{P_i\subseteq R_j} S_i$ for the index $j \in [d+1]$ maximizing $\sum_{P_i\subseteq R_j} w(S_i)$.

This finishes the description of the algorithm. We already argued that its running time is $2^{O_d(\sqrt{n})}$. We shall justify that it does output an independent set of weight at least a $\frac{1}{d+1}$ fraction of the optimum $\alpha(G)$.

***I* is indeed an independent set.** For any $j \in [d+1]$, consider two vertices $x, y \in \bigcup_{P_i\subseteq R_j} S_i$. If $\{x, y\} \in S_i$ for some i, then x and y are non-adjacent since S_i is an independent set of $G[P_i]$. Else $x \in S_i$ and $y \in S_{i'}$ for some $i \neq i'$. P_i and $P_{i'}$ are not linked by a black edge in $(G/\mathcal{P})[C_j]$ since R_j is an independent set in H_j, nor they can be linked by a red edge (there are none in $(G/\mathcal{P})[C_j]$). Thus again, x and y are non-adjacent in G.

***I* has weight at least $\frac{\alpha(G)}{d+1}$.** We claim that $\bigcup_{P_i\subseteq R_j} S_i$ is a heaviest independent set of $G[C_j]$. Note that the P_is that are included in C_j (and partition it) form a module partition of $G[C_j]$. In particular, any heaviest independent set intersecting some $P_i \subseteq C_j$ has to

[3] We use this notation as a slight abuse of notation for $(G/\mathcal{P})[\{P_i \,:\, P_i \subseteq C_j\}]$.

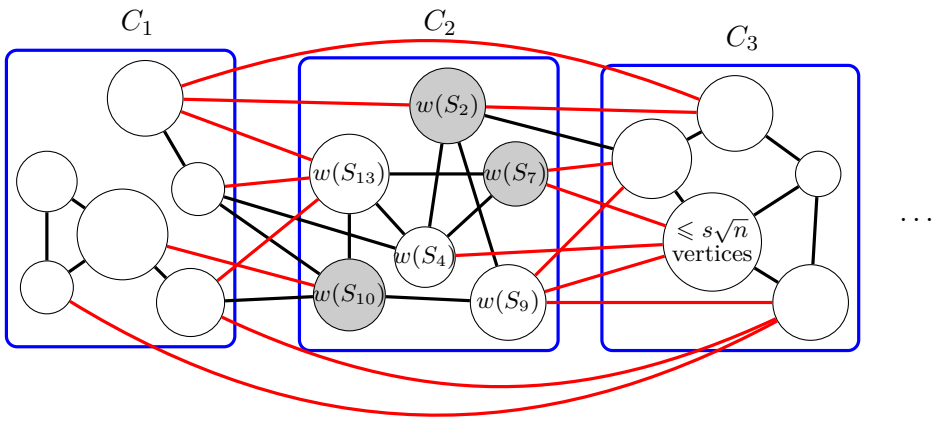


Figure 3.1: The trigraph $G/\mathcal{P}$ with its $\lfloor\sqrt{n}\rfloor$ vertices, each corresponding to a subset of at most $s\sqrt{n}$ vertices of G. The weights $w(S_i)$ of heaviest independent sets S_i of $G[P_i]$ for each part P_i of the color class C_2 of the $d+1$-coloring of $\mathcal{R}(G/\mathcal{P})$. A heaviest independent set in the so-weighted $(G/\mathcal{P})[C_2]$ (shaded) corresponds to an optimum solution in $G[\bigcup_{P_i\subseteq C_2} P_i]$. One of these $d+1$ independent sets is a $d+1$-approximation.

contain a heaviest independent of $G[P_i]$. This is precisely what the algorithm computes. Then a heaviest independent set in $G[C_j]$ packs such subsolutions to maximize the total weight, which is what is computed in H_j.

We conclude by the pigeonhole principle, since a heaviest independent set X of G is such that $w(X \cap C_j) \geqslant \frac{\alpha(G)}{d+1}$ for some $j \in [d+1]$. □

3.2.4 Improving the approximation factor

We notice in this short section that the approximation factor of lemma 76 can be improved using the notion of *clustered coloring*. The *clustered chromatic number* of a class of graphs is the smallest integer k such that there is a constant c for which all the graphs of the class can be k-colored such that every color class induces a subgraph whose connected components have size at most c. A proper coloring is a particular case of clustered coloring when $c = 1$.

Instead of properly coloring the red graph, as we did in the proof of lemma 76, we could use less colors and allow for small monochromatic components (in place of monochromatic components of size 1). We use for that the following bound due to Alon et al.

Theorem 77 ([73])**.** *The class of graphs of maximum degree at most d has clustered chromatic number at most $\lceil\frac{d+2}{3}\rceil$.*

We can use this lemma to improve our approximation algorithms.

Theorem 78. *On inputs as in lemma 76 with $s := 2^{4c_{d'}+4}$, and $d := c_{d'} \cdot 2^{4c_{d'}+4}$, Weighted Max Independent Set further admits an $\lceil\frac{d+2}{3}\rceil$-approximation algorithm in time $2^{O_d(\sqrt{n})}$.*

Proof. Again, we compute in polynomial time a partition $\mathcal{P} = \{P_1, \ldots, P_{\lfloor\sqrt{n}\rfloor}\}$ of $V(G)$ whose parts have size at most $s\sqrt{n}$ and such that $\mathcal{R}(G/\mathcal{P})$ has maximum degree at most d, using lemma 74. Let c be the constant such that $\mathcal{R}(G/\mathcal{P})$ admits a clustered coloring using $\lceil\frac{d+2}{3}\rceil$ colors such that each color class C_j (with $j \in [\lceil\frac{d+2}{3}\rceil]$) is such that the connected

components $C_j^1, C_j^2, \ldots, C_j^{h_j} \subseteq C_j$ of $\mathcal{R}(G/\mathcal{P})[C_j]$ have size at most c each. This coloring is guaranteed to *exist* by theorem 77. Due to the overall running time, we might as well compute it by exhaustive search, in time $2^{O_d(\sqrt{n})}$.

For every $j \in \lceil \frac{d+2}{3} \rceil$ and $h \in [h_j]$, we denote by $P_1(C_j^h), \ldots, P_{c(j,h)}(C_j^h)$ the $c(j,h) \leqslant c$ parts $P_i \in \mathcal{P}$ that are included in C_j^h. For every $j \in \lceil \frac{d+2}{3} \rceil$, every $h \in [h_j]$, and every $J \subseteq [c(j,h)]$, we compute a heaviest independent set in $G[\bigcup_{z \in J} P_z(C_j^h)]$, which we denote by $S_{j,h,J}$. This takes time $O(\sqrt{n} \cdot 2^c \cdot 2^{sc\sqrt{n}}) = 2^{O_d(\sqrt{n})}$ since $|\bigcup_{z \in J} P_z(C_j^h)| \leqslant c \cdot s\sqrt{n}$.

For each C_j, in time $(2^c)^{\sqrt{n}} = 2^{c\sqrt{n}}$, we exhaustively try all subsets $X \subseteq \bigcup_{P_i \in C_j} P_i$ that are unions of $S_{j,h,J}$ filtering them out when $G[X]$ is not edgeless, and keep a heaviest of them, say R_j. Since there can only be black edges or non-edges between some $P_i \in C_j^h$ and $P_{i'} \in C_j^{h'}$ with $h \neq h'$, it is clear that a heaviest independent set of $G[\bigcup_{P_i \in C_j} P_i]$ is indeed a union of $S_{j,h,J}$ (with fixed j). We output a heaviest set among the R_js, which is the desired $\lceil \frac{d+2}{3} \rceil$-approximation. The running time is as claimed. □

3.2.5 Time-approximation trade-offs

Lemma 76 and theorem 78 run exhaustive algorithms on induced subgraphs of size $O_d(\sqrt{n})$. As such, the latter inputs keep the same twin-width upper bound. To speed up the algorithm (admittedly while worsening the approximation factor) it is tempting to recursively call our very algorithm. We show that this leads to a time-approximation trade-off parameterized by an integer $q = 0, \ldots, O_d(\log \log n)$. At one end of this discrete curve, one finds the exact exponential algorithm ($q = 0$), and more interestingly the $d+1$-approximation in time $2^{O_d(\sqrt{n})}$ ($q = 1$), while at the other end lies a polynomial-time algorithm with approximation factor n^ε, where $\varepsilon > 0$ can be made as small as desired.

As we will deal with the same kind of recursions for several problems, we show the following generic abstraction.

Lemma 79. *Let $\hat{d}$ be a natural, $d' = 2\hat{d}+2$, and $d := c_{d'} \cdot 2^{4c_{d'}+4}$. Let Π be an optimization graph problem where inputs come with a $\hat{d}$-sequence of their n-vertex graph G, or with a neatly divided matrix $(M, (\mathcal{R}, \mathcal{C})) \in \mathcal{M}_{n,d'}$ conform to G. Let $\mathcal{P}$ be the partition of $V(G)$ given by lemma 75. Assume that*

1. *Π can be exactly solved in time $2^{O(n)}$, and there are constants c_1, c_2, c_3, and a function $f \geqslant 1$ such that*
2. *a $d^{c_3} r^2$-approximation of Π on G can be built in time n^{c_2} by using at most n^{c_1} calls to an r-approximation of Π—or another optimization problem Π' already satisfying the conclusion of the lemma—on an induced subgraph of G with at most $f(d)\sqrt{n}$ vertices or a full cleanup of an induced subtrigraph of $G/\mathcal{P}$ (on at most $\sqrt{n}$ vertices).*

Then Π can be $d^{c_3(2^q-1)}$-approximated in time

$$(f(d)^q n)^{(2-2^{-q})(c_1+c_2)} \cdot 2^{f(d)^{2(1-2^{-q})} n^{2^{-q}}},$$

for any non-negative integer q.

Proof. The proof is by induction on q. The case $q = 0$ is implied by item 1. The case $q = 1$, and the induction step in general, is nothing more than an abstraction of lemma 76, where exhaustive algorithms are replaced by recursive calls.

For any $q \geqslant 0$, we assume that Π can $d^{c_3(2^q-1)}$-approximated in the claimed running time, and show the same statement for the value $q+1$. Following item 2, we run this algorithm—or one for another optimization problem Π' satisfying the conclusion of the lemma—at most n^{c_1} times on $f(d)\sqrt{n}$-vertex induced subgraphs of the input graph G

or on full cleanups of induced subtrigraphs of $G/\mathcal{P}$. The latter graphs have at most $\sqrt{n} \leqslant f(d)\sqrt{n}$ vertices. By lemma 75, we can compute in polynomial time a neatly divided matrix $(M', (\mathcal{R}', \mathcal{C}')) \in \mathcal{M}_{|V(H)|,d'}$ conform to H, for each graph H of a recursive call; hence the induction applies.

Overall this takes time at most

$$\begin{aligned}
&n^{c_1} + n^{c_2} \cdot \left((f(d)^q \cdot f(d)\sqrt{n})^{(2-2^{-q})(c_1+c_2)} \cdot 2^{f(d)^{2(1-2^{-q})}(f(d)\sqrt{n})^{2^{-q}}} \right) \\
&\leqslant (f(d)^{q+1}n)^{c_1+c_2+\frac{1}{2}(2-2^{-q})(c_1+c_2)} \cdot 2^{f(d)^{2(1-2^{-q})+2^{-q}} n^{\frac{2^{-q}}{2}}} \\
&= (f(d)^{q+1}n)^{(2-\frac{2^{-q}}{2})(c_1+c_2)} \cdot 2^{f(d)^{2-2^{-q+1}+2^{-q}} n^{2^{-(q+1)}}} \\
&= (f(d)^{q+1}n)^{(2-2^{-(q+1)})(c_1+c_2)} \cdot 2^{f(d)^{2(1-2^{-(q+1)})} n^{2^{-(q+1)}}}.
\end{aligned}$$

For the first inequality, we assume that the two summands are larger than 2, so their sum can be bounded by their product.

Besides we get an approximation of factor at most $(d^{c_3(2^q-1)})^2 d^{c_3} = d^{c_3(2^{q+1}-1)}$. □

In more legible terms we have proved that:

Lemma 80. *Problems* Π *satisfying the assumptions of lemma 79 can be* $d^{O(1)(2^q-1)}$*-approximated in time* $2^{O_{d,q}(\sqrt[2^q]{n})}$*, for any non-negative integer* q*.*

If most graph problems admit single-exponential algorithms, we will deal with such a problem that is only known to be solvable in time $2^{O(n \log n)}$. Therefore we prove a variant of lemma 79 with a slightly worse running time.

Lemma 81. *Let* Π *be solvable in time* $2^{O(n \log n)}$ *and satisfy the second item of lemma 79. Then* Π *can be* $d^{c_3(2^q-1)}$*-approximated in time*

$$2^{\left((c_1+c_2)(2-2^{-q}) \log f(d) + f(d)^{2(1-2^{-q})} n^{2^{-q}}\right) \log n},$$

for any non-negative integer q*.*

Proof. We follow the proof of lemma 79 when the induction now gives a running time of

$$\begin{aligned}
&n^{c_2} + n^{c_1} \cdot 2^{\left((c_1+c_2)(2-2^{-q}) \log f(d) + (f(d)\sqrt{n})^{2^{-q}}\right) \log(f(d)\sqrt{n})} \\
&\leqslant 2^{\left((c_1+c_2)(2-2^{-(q+1)}) \log f(d) + f(d)^{2(1-2^{-(q+1)})} n^{2^{-(q+1)}}\right) \log n}.
\end{aligned}$$

□

Again the previous lemma can be rewritten as:

Lemma 82. *Problems* Π *satisfying the assumptions of lemma 81 can be* $d^{O(1)(2^q-1))}$*-approximated in time* $2^{O_{d,q}(\sqrt[2^q]{n} \log n)}$*, for any non-negative integer* q*.*

We derive from lemma 81 the following notable regimes.

Theorem 83. *Problems* Π *satisfying the assumptions of lemma 81 admit polynomial-time* n^{ε}*-approximation algorithms, for any* $\varepsilon > 0$*.*

Proof. This is the particular case $q = \lceil \log \frac{\varepsilon \log n}{2c_3 \log d} \rceil$.

Indeed the approximation factor is then at most $d^{c_3(2^q-1)} \leqslant d^{2c_3 \frac{\varepsilon \log n}{2c_3 \log d}} = 2^{\varepsilon \log n} = n^{\varepsilon}$, while the running time is at most

$$2^{\left((c_1+c_2)(2-2^{-q})\log f(d)+f(d)^{2(1-2^{-q})}n^{2^{-q}}\right)\log n} \leqslant 2^{\left(2(c_1+c_2)\log f(d)+f(d)^2 n^{\frac{2c_3 \log d}{\varepsilon \log n}}\right)\log n}$$

$$= n^{2(c_1+c_2)\log f(d)+f(d)^2 d^{\frac{2c_3}{\varepsilon}}}.$$

If further Π can be solved exactly in time $2^{O(n)}$ (hence satisfies the assumptions of lemma 79), one obtains a better running time, where the exponent of n does not depend on ε. Indeed,

$$(f(d)^q n)^{(2-2^{-q})(c_1+c_2)} 2^{f(d)^{2(1-2^{-q})}n^{2^{-q}}} \leqslant \left(\frac{\varepsilon \log n}{c_3 \log d}\right)^{2(c_1+c_2)\log f(d)} 2^{f(d)^2 d^{\frac{2c_3}{\varepsilon}}} n^{2(c_1+c_2)}. \quad \square$$

Theorem 84. *Problems Π satisfying the assumptions of lemma 79, resp. lemma 81, admit a $\log n$-approximation algorithm running in time $2^{O_d(n^{\frac{1}{\log \log n}})}$, resp. $2^{O_d(n^{\frac{1}{\log \log n}} \log n)}$.*

Proof. This is the particular case $q = \lfloor \log \left(\frac{\log \log n}{c_3 \log d} + 1 \right) \rfloor$.

This value is computed such that the approximation factor $d^{c_3(2^q-1)}$ is at most $\log n$. It can be easily checked that the running times are as announced. $\square$

We derive the following for Weighted Max Independent Set.

Theorem 85. *Weighted Max Independent Set on n-vertex graphs G (vertex-weighted by w) given with a d'-sequence satisfies the assumptions of lemma 79. In particular, this problem admits*

- *a $(d+1)^{2^q-1}$-approximation in time $2^{O_{d,q}(n^{2^{-q}})}$, for every integer $q \geqslant 0$,*
- *an n^{ε}-approximation in polynomial-time $O_{d,\varepsilon}(1) \log^{O_d(1)} n \cdot n^{O(1)}$, for any $\varepsilon > 0$, and*
- *a $\log n$-approximation in time $2^{O_d(n^{\frac{1}{\log \log n}})}$,*

with $d := c_{2d'+2} \cdot 2^{4c_{2d'+2}+4}$.

Proof. Even the exhaustive algorithm exactly solves WMIS in time $2^{O(n)}$. We thus focus on showing that WMIS satisfies the second item of lemma 79. We set $c_1 \geqslant 1$ as the required exponent to turn a d'-sequence into a neatly divided matrix of $\mathcal{M}_{n,2d'+2}$ conform to G, $c_2 = \frac{1}{2} + \eta$ for any fixed $\eta > 0$, the appropriate $1 < c_3 \leqslant 2$, and $f(d) = d \geqslant 1$.

The algorithm witnessing the second item is simply the proof of lemma 76. We first check that this algorithm makes $\lfloor \sqrt{n} \rfloor + d + 1$ recursive calls on induced subgraphs of the input G: each of the $\lfloor \sqrt{n} \rfloor$ graphs $G[P_i]$ where P_i has indeed size at most $O_d(\sqrt{n})$, and each of the $d+1$ graphs $(G/\mathcal{P})[C_j]$ (indeed an induced subgraph of G by definition of the black graph of a trigraph) on at most $\sqrt{n}$ vertices.

We finally assume that each recursive call outputs an r-approximation of WMIS. Let $j \in [d+1]$ be such that $w(C_j \cap I) \geqslant \frac{1}{d+1} w(I)$ for I a heaviest independent set of G vertex-weighted by w. Let $J \subseteq [\lfloor \sqrt{n} \rfloor]$ be the indices of the P_is that are intersected by $C_j \cap I$, that is, $J = \{i \ : \ P_i \cap (C_j \cap I) \neq \emptyset\}$. For every $i \in J$, set $w_i = w(P_i \cap I)$. Each recursive call on some P_i with $i \in J$, yields an independent set of weight at least $\frac{w_i}{r}$, by assumption. Thus the weights that our algorithm puts on $(G/\mathcal{P})[C_j]$ are such that it has an independent set of weight at least $\Sigma_{i \in J} \frac{w_i}{r} = \frac{w(C_j \cap I)}{r}$. As we run an r-approximation on this graph, we get an independent set of weight at least $\frac{w(C_j \cap I)}{r^2} \geqslant \frac{w(I)}{(d+1)r^2}$. Thus WMIS satisfies the assumptions of lemma 79, and we conclude. $\square$

3.3 Finding the suitable generalization: the case of Coloring

In this section, we deal with the Coloring problem. Unlike for WMIS, we cannot solely resort to recursively calling our Coloring algorithm on smaller graphs. The right problem generalization needs to be found for the inductive calls to work through, and it happens to be Set Coloring.

In the Set Coloring problem, the input is a couple (G, b) where G is a graph, and b is a function assigning a positive integer to each vertex of G. The goal is to find, for each $v \in V(G)$, a set S_v of at least $b(v)$ colors such that $S_u \cap S_v = \emptyset$ whenever $uv \in E(G)$, and minimizing $|\cup_{v \in V(G)} S_v|$. Let $\chi_b(G)$ be the optimal value of Set Coloring for (G, b). Observe that Coloring corresponds to the case where $b(v) = 1$ for every $v \in V(G)$.

Theorem 86. *Set Coloring (and hence Coloring) on n-vertex graphs G given with a d'-sequence satisfies the assumptions of lemma 81. In particular, this problem admits*

- *a $(d+1)^{2^q-1}$-approximation in time $2^{O_{d,q}(n^{2^{-q}} \log n)}$, for every integer $q \geqslant 0$, and*
- *an n^ε-approximation in polynomial-time for any $\varepsilon > 0$.*

with $d := c_{2d'+2} \cdot 2^{4c_{2d'+2}+4}$.

Proof. It is known [74] that Set Coloring can be solved using the inclusion-exclusion principle in time $O^*(\max_{v \in V(G)} b(v)^n) = 2^{O(n \log n)}$. We now prove that it satisfies the second item of lemma 79. We denote by $\mathcal{A}$ the r-approximation algorithm of the statement, which we will use on instances of Set Coloring. In particular, we will call it at most $\sqrt{n}+1$ times, and will obtain at the end a $(d+1)r^2$-approximation on our input (G, b) in polynomial time.

We first apply lemma 75 to get, in polynomial-time, a partition $\mathcal{P} = \{P_1, \ldots, P_{\lfloor\sqrt{n}\rfloor}\}$ of $V(G)$ whose parts have size at most $d\sqrt{n}$ and such that $R(G/\mathcal{P})$ has maximum degree at most d. For every $i \in [\lfloor\sqrt{n}\rfloor]$, we use $\mathcal{A}$ to compute an r-approximated solution c_{P_i} of $(G[P_i], b_{|P_i})$. We denote by b' the function which assigns, to each P_i, the number of colors of c_{P_i}. We now compute, in polynomial-time, a proper $(d+1)$-coloring of $R(G/\mathcal{P})$, which defines the sets $C_1, \ldots, C_{d+1}$. For each $j \in [d+1]$, we construct another Set Coloring instance consisting of the graph $H_j = (G/\mathcal{P})[C_j]$ (recall that this trigraph has no red edge, and can thus be seen as a graph), together with the function $b'_{|C_j}$. Again we use $\mathcal{A}$ to compute an r-approximated solution on $(H_j, b'_{|C_j})$. We denote by c_H this solution. Let G_j be the subgraph of G induced by $\cup_{P_i \in C_j} P_i$, and b_j the restriction of b to $V(G_j)$. We now show how to construct a solution c_j of Set Coloring to (G_j, b_j) from c_H and all c_{P_i}. Recall that for every $P_i \in C_j$, every $v \in P_i$, we have that $c_{P_i}(v)$ is a subset of $\{1, \ldots, b'(P_i)\}$ of size at least $b(v)$, and that $c_H(P_i)$ is a subset of size at least $b'(P_i)$. Hence, for each $P_i \in C_j$, one can choose an arbitrary bijection τ from $\{1, \ldots, b'(P_i)\}$ to $c_H(P_i)$, and define to each vertex $v \in P_i$ the set $c_j(v)$ as $\{\tau(x) : x \in c_{P_i}(v)\}$.

By construction, this solution is a feasible one for the instance (G_j, b_j). Let us prove that it is an r^2-approximation of $\chi_{b_j}(G_j)$. First, by definition of c_H, our solution uses at most $r \cdot \chi_{b'_{|C_j}}(H_j)$ colors. Then, by definition of c_{P_i} for every $P_i \in C_j$, we have $b'_{C_j}(P_i) \leqslant r \cdot \chi_{b_{|P_i}}(G[P_i])$. Now, denote by Γ the function which assigns to each $P_i \in C_j$ the number $\chi_{b_{|P_i}}(G[P_i])$. We now use the following claim, whose proof is left to the reader.

▷ Claim 87. Let (G, b) be an instance of Set Coloring, and $r \in \mathbb{R}_+$. It holds that $\chi_{r \cdot b}(G) \leqslant r \cdot \chi_b(G)$, where $r \cdot b$ is the function which assigns $r \cdot b(v)$ to each $v \in V(G)$.

This implies $\chi_{b'_{|C_j}}(H_j) \leqslant r \cdot \chi_\Gamma(H_j)$, and thus our solution uses at most $r^2 \cdot \chi_\Gamma(H_j)$ colors. We now prove the following claim.

▷ Claim 88. $\chi_\Gamma(H_j) \leqslant \chi_{b_j}(G_j)$.

Proof of the claim. Let c be an optimal solution for (G_j, b_j). For every distinct $P_i, P_{i'} \in C_j$ such that $P_iP_{i'}$ is an edge of H_j, it holds that there are all possible edges between P_i and $P_{i'}$ in G_j (by definition of the coloring $C_1, \cdots, C_{d+1}$), hence it holds that $\bigcup_{v\in P_i} c(v)$ and $\bigcup_{v\in P_{i'}} c(v)$ have empty intersection. Moreover, by definition of Γ, we have that $\bigcup_{v\in P_i} c(v)$ is of size at least $\Gamma(P_i)$, hence the function which assigns $\bigcup_{v\in P_i} c(v)$ to each P_i is a feasible solution for (H_j, Γ) using at most $\chi_{b_j}(G_j)$ colors. □

We now have in hand an r^2-approximated solution of (G_j, b_j) for every $j \in [d+1]$, which can be turned into a $(d+1)r^2$-approximated solution of (G, b), as desired. □

3.4 Edge-based problems: the case of Max Induced Matching

So far, we only considered problems where approximated solutions in each part P_i of a partition $\mathcal{P}$ of $V(G)$ of small width, and in some selected induced subgraphs of $(V(G/\mathcal{P}), E(G/\mathcal{P}))$, were enough to build an approximated solution for G.[4] We now handle problems for which a number of edges is to be optimized. Now all competitive solutions can integrally lie in between pairs of parts P_i, P_j linked by a black or a red edge in $G/\mathcal{P}$. This complicates matters, and forces us to be competitive there as well, naturally splitting the algorithm into three subroutines.

We present the algorithms for Max Subset Induced Matching where one is given, in addition to the input graph G (possibly with edge weights), a subset $Y \subseteq E(G)$, and the goal is to find a heaviest induced matching S of G such that $S \subseteq Y$. Then Max Induced Matching is the particular case when $Y = E(G)$. Of course, we could solely use the edge weights to emulate Y (by giving negative weights to all the edges in $E(G) \setminus Y$). We believe this formalism is slightly more convenient for the reader to quickly and explicitly identify where our algorithm is seeking mutually induced edges.

Since the case of Max Induced Matching is more involved than were the treatment of MIS and Coloring, we again split the arguments into the design of a subexponential-time constant-approximation algorithm (lemma 89) followed by how this algorithm meets the requirements of lemma 79 (theorem 90).

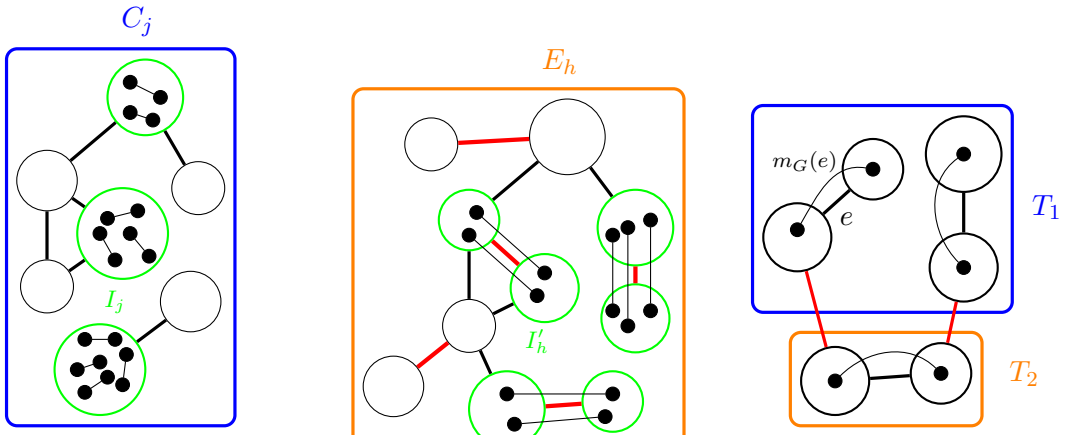
Lemma 89. *Assume every input (G, Y) is given with a d'-sequence of the n-vertex, edge-weighted by w, graph G. We set $d := c_{d'} \cdot 2^{4c_{d'}+4}$, and $s := 2^{4c_{d'}+4}$. Max Subset Induced Matching can be $O(d^2)$-approximated in time $2^{O_d(\sqrt{n})}$ on these inputs.*

Proof. Again, by lemma 74, we start by computing in polynomial time a partition of $V(G)$, $\mathcal{P} = \{P_1, \ldots, P_{\lfloor\sqrt{n}\rfloor}\}$, of parts with size at most $s\sqrt{n}$ and such that $\mathcal{R}(G/\mathcal{P})$ has maximum degree at most d.

We $(d+1)$-color $\mathcal{R}(G/\mathcal{P})$, which defines a coarsening $\{C_1, \ldots, C_{d+1}\}$ of $\mathcal{P}$. We also distance-2-edge-color $\mathcal{R}(G/\mathcal{P})$ with $z = 2(d-1)d + 1$ colors, that is, properly (vertex-)color the square of its line graph. Observe that $z - 1$ upperbounds the maximum degree of the square of the line graph of $\mathcal{R}(G/\mathcal{P})$. This partitions the edges of $\mathcal{R}(G/\mathcal{P})$ into $\{E_1, \ldots, E_z\}$. For each red edge $e = P_iP_j \in R(G/\mathcal{P})$, we denote by $p(e)$ the set $P_i \cup P_j$. We also set $X_h = p(E_h) = \bigcup_{e\in E_h} p(e)$ for each $h \in [z]$.

Let $M \subseteq Y$ be a fixed (unknown) heaviest induced matching of G contained in Y. Let M_v, M_r, M_b partition M, where M_v (as **v**ertex) consists of the edges of M with both

[4] The improvement based on clustered coloring slightly departed from that simple scheme.



(a) Computing R_j consists first of determining the heaviest induced matching in each part P_i and then, for color C_j, to compute the maximum independent set I_j (in green) weighted by the size of the matchings.

(b) Color E_h reveals a set of red edges from trigraph $G/\mathcal{P}$. Set R'_h corresponds to the heaviest matching among these edges which is mutually induced regarding the black edges. The weight of the red edges e is $w(S'_e)$.

(c) An example of set S of size 3 with two colors T_1 and T_2. The induced matching R''_i for color T_i is obtained by considering the maximum-weighted edge $m_G(e)$ between the two parts of e.

Figure 3.2: Illustration of how to determine the induced matching N_v, N_r, and N_b (in that order, from left to right).

endpoints in a same P_i, M_r (as **r**ed) corresponds to edges of M between some P_i and P_j with $P_iP_j \in R(G/\mathcal{P})$, and M_b (as **b**lack), the edges of M between some P_i and P_j with $P_iP_j \in E(G/\mathcal{P})$. We compute three induced matchings $N_v, N_r, N_e \subseteq Y$ of G, capturing a positive fraction of M_v, M_r, M_e, respectively. fig. 3.2 gives the intuition of the procedures which determine each of these approximated solutions.

Computing N_v. For every integer $1 \leqslant i \leqslant \lceil\sqrt{n}\rceil$, we compute a heaviest induced matching in $G[P_i]$ contained in Y, say S_i, in time $2^{O_d(\sqrt{n})}$. For each $j \in [d+1]$, let H_j be the *graph* $(G/\mathcal{P})[C_j]$ with every vertex $P_i \in C_j$ weighted by $w(S_i)$. We compute a heaviest independent set I_j in H_j, also in time $2^{O_d(\sqrt{n})}$.

Let R_j be the induced matching $\{e \in S_i : P_i \in I_j\}$. It is indeed an induced matching in G contained in Y, since each S_i is so, there is no red edge in $(G/\mathcal{P})[C_j]$, and I_j is an independent set of H_j. The solution N_v is then a heaviest among the R_js.

Computing N_r. For each $e = P_iP_j \in R(G/\mathcal{P})$, we compute a heaviest induced matching S'_e in $G[p(e)] = G[P_i \cup P_j]$ among those that are included in Y and have only edges with one endpoint in P_i and the other endpoint in P_j. This takes times at most $\frac{\sqrt{n}d}{2} \cdot 2^{O_d(\sqrt{n})} = 2^{O_d(\sqrt{n})}$ by trying out all vertex subsets, since $|P_i \cup P_j| \leqslant 2s\sqrt{n}$. For each $h \in [z]$, let H'_h be the *graph* $(G/\mathcal{P})[\{P_i \ : \ P_i \text{ is incident to an edge } e \in E_h\}]$ and the red edges $e \in E_h$ are turned black and get weight $w(S'_e)$. We compute a heaviest induced matching I'_h in H'_h among those *included in* E_h, in time $2^{O_d(\sqrt{n})}$. Note here that we changed the prescribed set of edges Y to E_h.

Let R'_h be the induced matching $\{f \in S'_e : e \in I'_h\} \subseteq Y$ of G. Indeed, each $S'_e \subseteq Y$ is an induced matching, and there is no red edge between an endpoint of $e \in I'_h$ and an endpoint of $e' \neq e \in I'_h$ (since E_h is a color class in a distance-2-edge-coloring of $\mathcal{R}(G/\mathcal{P})$), nor a black edge (by virtue of I'_h being an induced matching of H'_h). The solution N_r is then a heaviest among the R'_hs.

Computing N_b. Observe first that an induced matching of G can only contain at most one edge between P_i and P_j when $P_iP_j \in E(G/\mathcal{P})$. Thus in the graph $(V(G/\mathcal{P}), E(G/\mathcal{P}))$, we give weight $\max\{w(f) : f = uv \in Y, u \in P_i, v \in P_j\}$, with the convention that $\max \emptyset = -1$, to each edge $e = P_iP_j \in E(G/\mathcal{P})$, call G' the resulting edge-weighted graph, and denote by $m_G(e)$ an edge $f \in Y$ realizing this maximum. We compute a heaviest induced matching S of G' *included in* $E(G')$, in time $2^{O_d(\sqrt{n})}$. Let H_S be the graph with vertex set S, and an edge between e and e' whenever there is a red edge in $G/\mathcal{P}$ between an endpoint of e and an endpoint of e'. As H_S has degree at most $2d$, it can be $2d+1$-colored; let $T_1, \ldots, T_{2d+1}$ the corresponding color classes.

For each $i \in [2d+1]$, let R''_i be the induced matching $\{m_G(e) : e \in T_i\} \subseteq Y$ of G. Indeed, S is an induced matching in the black graph of $G/\mathcal{P}$, and the underlying vertices of T_i do not induce any red edge in $G/\mathcal{P}$, by design. The solution N_r is then a heaviest among the R''_is.

We finally output a heaviest set among N_v, N_r, N_b. The overall running time is $2^{O_d(\sqrt{n})}$ as we make a polynomial number of calls to (exhaustive) subroutines on graphs with $O_d(\sqrt{n})$ vertices, and color in linear time $O(n)$-vertex graphs of maximum degree Δ with $\Delta + 1$ colors. We already argued that $N_v, N_r, N_b \subseteq Y$ are all induced matchings in G, thus so is our output.

We shall just show that we meet the claimed approximation factor. First, one can observe $w(N_v) \geqslant \frac{w(M_v)}{d+1}$. Second, at least a $\frac{1}{z}$ fraction of the weight of M_r intersects some fixed E_i (with $i \in [z]$). Let $\mathcal{J}$ be the parts of $\mathcal{P}$ intersected by $M_r \cap X_i$. As there cannot be a black edge between two parts of $\mathcal{J}$ (otherwise M_r is not an induced matching as defined), our algorithm indeed computes an induced matching of $G[X_i]$ included in Y of weight at least $w(M_r \cap X_i)$. Hence $w(N_r) \geqslant \frac{w(M_r)}{z}$.

Third, we already argued that an induced matching in G' corresponds to an induced matching in the black graph of $G/\mathcal{P}$. Thus at least one of the R''_i (with $i \in [2d+1]$) contains at least a $\frac{1}{2d+1}$ fraction of the weight of M_b. Therefore $w(N_b) \geqslant \frac{w(M_b)}{2d+1}$.

Finally the output induced matching has at least weight

$$\frac{w(M)}{3 \cdot \max(d+1, z, 2d+1)} = \frac{w(M)}{3z} = \frac{w(M)}{3(2(d-1)d+1)}.$$ □

Theorem 90. *Max Subset Induced Matching on an n-vertex graph G, edge-weighted by w, with prescribed set $Y \subseteq E(G)$, and given with a d'-sequence, satisfies the assumptions of lemma 79. In particular, this problem admits*

- *a $(d+1)^{2^q-1}$-approximation in time $2^{O_{d,q}(n^{2^{-q}})}$, for every integer $q \geqslant 0$,*
- *an n^{ε}-approximation in polynomial-time $O_{d,\varepsilon}(1) \log^{O_d(1)} n \cdot n^{O(1)}$, for any $\varepsilon > 0$, and*
- *a $\log n$-approximation in time $2^{O_d(n^{\frac{1}{\log \log n}})}$,*

with $d := c_{2d'+2} \cdot 2^{4c_{2d'+2}+4}$.

Proof. The exhaustive algorithm (trying out all vertex subsets and checking whether they induce a matching included in Y) solves Max Subset Induced Matching in time $2^{O(n)}$. Thus we show Max Subset Induced Matching satisfies the second item of lemma 79, as witnessed by lemma 89 where subcalls are dealt with recursively. We set $c_2 \geqslant 1$ as the required exponent to turn a d'-sequence into a neatly divided matrix of $\mathcal{M}_{n,2d'+2}$, and compute the various needed colorings, the appropriate $\frac{1}{2} < c_1 < 1$, and $2 < c_3 < 3$, and $f(d) = 2d \geqslant 1$ with $s := 2^{4c_{d'}+4}$.

In computing N_v, the algorithm makes $\lfloor \sqrt{n} \rfloor$ recursive calls and $d+1$ calls to Weighted Max Independent Set on induced subgraphs of G. All of these induced subgraphs are

on less than $f(d)\sqrt{n}$ vertices. Computing N_r makes at most $\frac{\sqrt{n}d}{2}$ recursive calls on induced subgraphs of G with at most $f(d)\sqrt{n}$ vertices, followed by at most $2(d-1)d+1$ recursive calls on full cleanups of induced subtrigraphs of $G/\mathcal{P}$ with at most $\sqrt{n}$ vertices (in fact, one can observe that the latter recursive calls happen to also be on induced subgraphs of G). Finally, computing N_b makes one recursive call to a full cleanup of $G/\mathcal{P}$ on $\lfloor\sqrt{n}\rfloor$ vertices.

In summary, we make $O_d(\sqrt{n})$ recursive calls or calls to another problem WMIS (which already satisfies lemma 79 with better constants) on induced subgraphs of G or full cleanups of (the whole) $G/\mathcal{P}$, each on $O_d(\sqrt{n})$ vertices. Hence, by lemma 75, the induction applies.

We check that getting r-approximations on every subcall allows to output a global $3(2(d-1)d+1)r^2$-approximation. For that we argue that N_v (resp., N_r, N_b) is a $(2(d-1)d+1)r^2$-*approximation* of M_v (resp., M_r, M_b). The fact that N_v is a $(d+1)r^2$-approximation (hence a $(2(d-1)d+1)r^2$-approximation, since we assume that $d \geqslant 1$) of M_v directly follows theorem 85.

We now show that N_r is a $(2(d-1)d+1)r^2$-approximation of M_r. Let $h \in [z] = [2(d-1)d+1]$ be an index maximizing $w(M_r \cap E(G[X_h]))$. Thus $w(M_r \cap E(G[X_h])) \geqslant \frac{w(M_r)}{2(d-1)d+1}$. Let $F_h \subseteq E_h$ be the edges $e = P_iP_j$ of $\mathcal{R}(G/\mathcal{P})$ that are inhabited by M_r (i.e., M_r contains at least one edge between P_i and P_j). Note that our algorithm makes an r-approximation of the optimum such solutions on $p(e)$ (selecting only edges between P_i and P_j). Thus the r-approximation on H'_h yields the desired $(2(d-1)d+1)r^2$-approximation N_r.

Finally, one can easily see that N_b is a $(2d+1)r$-approximation of M_b (note, here, the absence of a 2 in the exponent of r). □

3.5 Technical generalizations

3.5.1 Mutually Induced H-packing

In this section we present a far-reaching generalization of the approximation algorithms for Max Independent Set and Max Induced Matching. For any fixed graph H, let Mutually Induced H-packing be the problem where one seeks a largest collection of mutually induced copies of H in the input graph G, that is, a largest set S such that $G[S]$ is a disjoint union of (copies of) graphs H. We get similar approximation guarantees for Mutually Induced H-packing, for any connected graph H. Observe that Max Independent Set and Max Induced Matching are the special cases when H is a single vertex and a single edge, respectively.

We in fact approximate a technical generalization that we call Annotated Mutually Induced H-packing. The input is a tuple $(G, w, z, \gamma, \gamma')$ where G is a graph, $w : V(G)^{|V(H)|} \to \mathbb{Q}$ is a weight function over the tuples *without repetition* of $V(G)$ of size $|V(H)|$ (that we will use to keep track of the number of mutually induced copies *within* a given tuple of vertices of G), z is an integer between 1 and $|V(H)|$, $\gamma : V(G) \to [z]$ is a labeled partition of $V(G)$ into z classes, and $\gamma' : V(H) \to [z]$ is a labeled partition of $V(H)$ into z classes. Note that the Mutually Induced H-packing is obtained when $w(Z) = [G[Z]$ is isomorphic to $H]$ (where [.] is the Iverson bracket, i.e., taking value 1 if the property it surrounds is true, and 0 otherwise) and $z = 1$ (which forces the value of γ and γ'). The goal is to find a subset S such that

- $G[S]$ is a disjoint union of copies of H,
- there is an isomorphism between each copy C of H (in S) and H which preserves γ, γ', i.e., every vertex v of C is mapped to a vertex $v' \in V(H)$ with $\gamma(v) = \gamma'(v')$,

and

- $\sum_{C \text{ copy of } H \text{ in } S} w(V(C))$ is maximized.

We will need the notion of *compatible trigraphs* of a (labeled) graph. Given a graph H, we call *compatible trigraph of* H any trigraph on at most $|V(H)|$ vertices obtained by turning some (possibly none) black edges or non-edges of trigraph $H/\mathcal{Q}$ (for any fixed choice of a partition $\mathcal{Q}$ of $V(H)$) into red edges. In other words, a compatible trigraph H' of H is such that there is a cleanup H'' of H' that is also a quotient trigraph of H. Note that the number of compatible trigraphs of an h-vertex graph H is upperbounded by $B_h \cdot 2^{\binom{h}{2}} = 2^{O(h^2)}$, where B_h is the h-th Bell number, which counts the number of partitions of a set of size h.

Given a graph G vertex-partitioned by $\mathcal{P}$ and a trigraph H, a subset $S \subseteq V(G)$ is said *cut by* $\mathcal{P}$ *along* H if $G[S]/\mathcal{P}$ is isomorphic to H. By extension, the copy of $G[S]$ in G (induced by S) is also said cut by $\mathcal{P}$ along H.

Lemma 91. *For any connected graph H, Annotated Mutually Induced H-packing, when every input $(G, w, z, \gamma, \gamma')$ is given with a d'-sequence of the n-vertex graph G, satisfies the assumptions of lemma 79. In particular, this problem admits*

- *a $d^{O_h(2^q)}$-approximation in time $2^{O_{d,h,q}(n^{2^{-q}})}$, for every integer $q \geqslant 0$,*
- *an n^{ε}-approximation in polynomial-time $O_{\varepsilon}(1) \cdot n^{O_{d,h}(1)}$, for any $\varepsilon > 0$,*

with $h = |V(H)|$, and $d := c_{2d'+2} \cdot 2^{4c_{2d'+2}+4}$.

Proof. As the first item of lemma 79 is satisfied, we describe an algorithm that fulfills the requirement of its second item. We proceed by induction on the number of vertices of H. Thus we can assume that Annotated Mutually Induced J-packing, with J a connected graph on less vertices than H, satisfies lemma 79. We already did the base case of the induction, which was Weighted Max Independent Set.

Algorithm. Again, by lemma 75, we start by computing in polynomial time a partition of $V(G)$, $\mathcal{P} = \{P_1, \ldots, P_{\lfloor\sqrt{n}\rfloor}\}$, of parts with size at most $d\sqrt{n}$ and such that $\mathcal{R}(G/\mathcal{P})$ has maximum degree at most d. Let S be a fixed (unknown) heaviest (with respect to w) mutually induced H-Packing of G preserving γ, γ'.

For every compatible trigraph H' of H, we look for mutually induced copies of H in G cut by $\mathcal{P}$ along H', and preserving γ, γ'. As the number of compatible trigraphs of H is $2^{O(h^2)}$, a $1/2^{O(h^2)}$ fraction of the weight of S is made of mutually induced copies of H which are cut by $\mathcal{P}$ along a fixed compatible trigraph H'. We now focus on this particular "run."

We distinguish two cases:

- (A) H' has at least one black edge, or
- (B) H' has no black edge.

As H is connected, the total graph of H' is also connected. Indeed, switching some edges or non-edge to red edges in the quotient trigraph of H cannot disconnect the total graph, which can only gain edges. Thus in case (A), every red component of H' has at least one incident black edge, and in case (B), H' has a single red component (and no black edge).

In general, we want to individually pack red components of H' (first type of recursive calls in smaller induced subgraphs of G), then combine those red components by connecting them with the right pattern of black edges (second type of recursive calls in the total graph of $G/\mathcal{P}$). Handling both cases (A) and (B) in an unified way runs into the technical issue that the weight function may destroy our combined solutions in an uncontrollable manner. The case distinction works as a win-win argument. In case (A), due to the presence of a black edge in H', we can pack at most one mutually induced copy of H within any

fixed subtrigraph of $G/\mathcal{P}$ matching H'. We thus exempt ourselves from the first type of recursive calls. In case (B), we do need the two types of recursive calls (as in WMIS), but the first type is done on the whole H. Thus the current weight function (on h-tuples) is informative enough.

Case (A). The essential element here is to build a new weight function w' on the h'-tuples of the total graph $\mathcal{T}(G/\mathcal{P})$, with $h' := |V(H')|$. For every injective map $\iota : V(H') \to \mathcal{P}$ inducing a trigraph isomorphism and preserving γ, γ', for every ordering of $\iota(V(H'))$ into an h'-tuple $(P_1, \ldots, P_{h'})$, we set

$$w'(P_1, \ldots, P_{h'}) := \max\{w(v_1^1, v_1^2, \ldots, v_1^{a_1}, \ldots, v_{h'}^1, v_{h'}^2, \ldots, v_{h'}^{a_{h'}}) \ : \ v_1^1, v_1^2, \ldots, v_1^{a_1} \in P_1, \ldots$$

$$v_{h'}^1, v_{h'}^2, \ldots, v_{h'}^{a_{h'}} \in P_{h'}, \text{ and } G[\{v_1^1, v_1^2, \ldots, v_1^{a_1}, \ldots, v_{h'}^1, v_{h'}^2, \ldots, v_{h'}^{a_{h'}}\}] \text{ is isomorphic to } H\}.$$

Indeed as we previously observed, in case (A), at most one mutually induced copy of H respecting the cut along H' can be packed in the subgraph of G induced by the vertices of $\iota(V(H'))$. (In the definition of w', we can further impose that a_i matches the number of vertices of H in the corresponding part of H' but this is not necessary.)

All the h'-tuples not getting an image by w' in the previous loop (realized in time $n^{O(h)}$) are assigned the value 0. We then make a recursive call to Annotated Mutually Induced $\mathcal{T}(H')$-packing on input $(\mathcal{T}(G/\mathcal{P}), w', 1, \gamma_0, \gamma_0')$ where we recall that $\mathcal{T}(.)$ is the total graph, and γ_0, γ_0' are the constant 1 functions.

Case (B). For every injective map $\iota : V(H') \to \mathcal{P}$ inducing a trigraph isomorphism and preserving γ, γ', we make a recursive call to Annotated Mutually Induced H-packing with input $(G_\iota = G[\bigcup_{P \in \iota(V(H'))} P], w, h, \gamma_\iota, \gamma_\iota')$ where two vertices get the same label by γ_ι if and only if they have the same label by γ and lie in the same $P \in \iota(V(H'))$, and γ_ι' gives to a vertex $v' \in X \in V(H')$ of H the same label given to the vertices $v \in \iota(X)$ such that $\gamma'(v') = \gamma(v)$. Informally $\gamma_\iota, \gamma_\iota'$ forces the recursive call to commit to the map ι and the former functions γ, γ'.

Each such recursive call yields a mutually induced packing of H. Since the red graph of $G/\mathcal{P}$ has degree at most d, we can color the (ordered) tuples of $\mathcal{P}$ of length up to h and inducing a connected subgraph of $\mathcal{R}(G/\mathcal{P})$ with at most $p(h,d) = hd^{2h} \cdot d^{2h} \cdot h! + 1$ colors such that every color class consists of disjoint tuples pairwise not linked by a red edge in $G/\mathcal{P}$. Indeed the claimed number of colors minus 1 upperbounds, in $\mathcal{R}(G/\mathcal{P})$, the number of connected tuples of length up to h that can touch (i.e., intersect or be adjacent to) a fixed connected tuple of length up to h. One color class contains a fraction $1/p(h,d)$ of the weight of the optimal solution S (subject to the same constraints). Running through all color classes j (and focusing on one containing a largest fraction of the optimum), we define a weight function w' on the h'-tuples of $\mathcal{T}(G\mathcal{P})$, with $h' = |V(H')|$, by giving to a tuple the weight returned by the corresponding recursive call whenever it is part of color class j, and weight 0 otherwise. We then make a recursive call to Annotated Mutually Induced $\mathcal{T}(H')$-packing on input $(\mathcal{T}(G/\mathcal{P}), w', 1, \gamma_0, \gamma_0')$ where we recall that $\mathcal{T}(.)$ is the total graph, and γ_0, γ_0' are the constant 1 functions.

We output a heaviest solution among all runs. We now check that the algorithm is as prescribed by lemma 79.

Number of recursive calls. We make at most $2^{O(h^2)} \cdot h \cdot |V(G/\mathcal{P})|^h = n^{O_h(1)}$ recursive calls *to Annotated Mutually Induced H-packing*, and at most $p(h,d)+1 = O_{d,h}(1)$ recursive calls *to Annotated Mutually Induced $\mathcal{T}(H')$-packing*. Hence there is a constant c_1 (function of d and h) such that the number of calls is bounded by n^{c_1}.

Nature and size of the inputs of the recursive calls. Both H and $\mathcal{T}(H')$ have strictly less vertices than H or are equal to H. Thus the induction on h applies. Besides,

$G[\bigcup_{P\in\iota(V'(H))} P]$ is an induced subgraph of G of size at most $h \cdot d\sqrt{n} = O_{d,h}(1) \cdot \sqrt{n}$, and $\mathcal{T}(G/\mathcal{P})$ is a full cleanup of $G/\mathcal{P}$ of size at most $\lfloor\sqrt{n}\rfloor$.

Running time. Outside of the recursive calls, one can observe that our algorithm takes times $O_{d,h}(1) \cdot n^{O_h(1)}$. Hence there is a constant c_2 (function of d and h) such that the running time of that part is bounded by n^{c_2}.

Correctness and approximation guarantee. As all the recursive calls are on induced subgraphs of G or of the total graph $\mathcal{T}(G/\mathcal{P})$, we return a mutually induced collection of graphs of the size of H. All these graphs are indeed induced copies of H since the weight function prevents the false positives of copies of H in the total graph $\mathcal{T}(G/\mathcal{P})$ but not in G (these tuples are given weight 0). Finally it can be checked that the returned solution has weight a fraction $(2^{O(h^2)} \cdot \max(r, p(h,d)r^2))^{-1}$ of the optimum, which can also be seen as a $d^{c_3}r^2$-approximation for some constant c_3 depending on d and h. □

3.5.2 Independent induced packing of stars and forests

The techniques employed to design approximations algorithms for Max Subset Induced Matching can be extended in order to tackle more general problems. In particular, we show in this section a generalization of theorem 90 for Max Edge Induced Star Forest and Max Edge Induced Forest. These two problems stand as the version of Mutually Induced $\mathcal{H}$-packing where $\mathcal{H}$ is respectively either the infinite family of stars or trees.

On the one hand, Max Edge Induced Star Forest asks, given a graph G and a subset $Y \subseteq E(G)$, for a collection of induced stars on G, made up of edges of Y only, maximizing the number of edges (or leaves).

Max Edge Induced Star Forest
Input: Graph G, subset $Y \subseteq E(G)$
Output: Collection $(A_i)_{i\in[k]}$ of induced stars on G, made up of edges in Y only, such that there is no edge between A_i and A_j, for any $i \neq j \in [k]$, which maximizes the number of edges.

On the other hand, given the same input, Max Edge Induced Forest asks for an induced forest F on G with the largest set of edges.

We would like to emphasize the fact that the objective function of both problems counts the number of *edges* in the solution, instead of *vertices*, as it is often the case in the literature when looking for a collection of stars or trees in a graph. The reason for this is because an approximated solution for these vertex versions can be obtained from an approximated solution of Weighted Max Independent Set (since any independent set is a star forest, and any forest is a bipartite graph).

Observe moreover that a solution of Max Edge Induced Forest can be 3-approximated with a solution of Max Edge Induced Star Forest. Indeed, the edge set of any tree can be partitioned into three distance-2-edge colors, which consist of a collection of stars. Therefore, the induced forest F can be partitioned into three collections of induced stars. In the remainder, we design approximation algorithms for Max Edge Induced Star Forest, and directly deduce results for Max Edge Induced Forest.

In the remainder, we propose approximation algorithms for Max Edge Induced Star Forest. We provide in particular a n^{ε}-approximation algorithm for Max Edge Induced Star Forest, running in polynomial time.

We need to find the suitable generalization of Max Edge Induced Star Forest, as it was done for Coloring in section 3.3. We call this problem Max Leaves Induced

Star Forest. Now, a weight function on vertices is added to the input, and we seek a collection of mutually induced stars with maximum weight, the weight of a star being the sum of the weights of its leaves (that is, the weight of the root is omitted).

Max Leaves Induced Star Forest
Input: Graph G, weights $w_V : V \to \mathbb{N}$, subset $Y \subseteq E(G)$
Output: Collection $(A_i)_{i\in[k]}$ of induced stars on G with root r_i, $A_i = \{r_i, s_i^1, \ldots, s_i^{L_i}\}$, made up of edges in Y only, with no edge between A_i and A_j, for any $i \neq j \in [k]$, maximizing

$$\sum_{i=1}^{k} w_V(A_i) = \sum_{i=1}^{k}\sum_{\ell=1}^{L_i} w(s_i^\ell)$$

We prove that Max Leaves Induced Star Forest follows the framework proposed in lemma 79. We begin with the design of a subexponential-time algorithm approximating a solution of Max Leaves Induced Star Forest with a ratio function of twin-width.

Lemma 92. *Assume every input of Max Leaves Induced Star Forest is given with a d'-sequence of the n-vertex G, and $d := c_{2d'+2} \cdot 2^{4c_{2d'+2}+4}$. Max Leaves Induced Star Forest can be $O(d^2)$-approximated in time $2^{O_d(\sqrt{n})}$ on these inputs.*

Proof. We compute in polynomial time a partition of $V(G)$, $\mathcal{P} = \{P_1, \ldots, P_{\lfloor\sqrt{n}\rfloor}\}$, of parts with size at most $d\sqrt{n}$ and such that $\mathcal{R}(G/\mathcal{P})$ has maximum degree at most d, by lemma 74.

As in lemma 76, we $(d+1)$-color $\mathcal{R}(G/\mathcal{P})$, which defines a coarsening $\{C_1, \ldots, C_{d+1}\}$ of $\mathcal{P}$. Moreover, we distance-2-edge-color $\mathcal{R}(G/\mathcal{P})$ with $z = 2(d-1)d+1$ colors. This partitions the edges of $\mathcal{R}(G/\mathcal{P})$ into $\{E_1, \ldots, E_z\}$. For each red edge $e = P_iP_j \in R(G/\mathcal{P})$, we denote by $p(e)$ the set $P_i \cup P_j$.

Let $A = \bigcup_{i=1}^{k} A_i$ be the union of all stars present in an optimum solution of Max Leaves Induced Star Forest in G. We have $A \subseteq Y$. Let A_v, A_r, A_b partition A, where A_v contains the edges of A with both endpoints in a same P_i, A_r corresponds to edges of A between some P_i and P_j with $P_iP_j \in R(G/\mathcal{P})$, and A_b, the edges of A between some P_i and P_j with $P_iP_j \in E(G/\mathcal{P})$. The set of edges A_v (resp. A_r, A_b) still form a collection of mutually induced stars. At least one over the three solutions produced by the partition A_v, A_r, A_b gives us a 3-approximation for this problem. Our algorithm consists of computing three solutions for Max Leaves Induced Star Forest of G, capturing a positive fraction of A_v, A_r, A_b, respectively.

Computing a $d+1$-approx for A_v. *Construction.* For every integer $1 \leqslant i \leqslant \lceil\sqrt{n}\rceil$, we compute an optimum solution for Max Leaves Induced Star Forest in $G[P_i]$ contained in Y, say S_i, in time $2^{O_d(\sqrt{n})}$. This can be achieved with guesses of the vertices in P_i, as $|P_i| \leqslant d\sqrt{n}$.

Then, we focus on each color C_j of $\mathcal{R}(G/\mathcal{P})$, for $j \in [d+1]$. There is no red edge in $H_j = (G/\mathcal{P})[C_j]$. We compute a heaviest independent set I_j in H_j where the parts P_i are weighted by the edge weight of S_i. Let R_j be the union of all optimum solutions for Max Leaves Induced Star Forest on all P_i belonging to I_j. The solution returned is the maximum over all R_js.

Approximation ratio. Let A_v^j be the subset of A_v made up of edges belonging to parts of C_j. There is no red edge between two parts of C_j, therefore their neighborhood consists of either full adjacency or full non-adjacency. As a consequence, a maximum-weighted collection of stars in C_j with edges inside parts intersects parts which are pairwise non-adjacent in $(G/\mathcal{P})[C_j]$, otherwise the stars are not mutually induced. Consequently, this justifies that the set R_j returned for each C_j is a maximum-weighted collection of stars

in C_j made up of edges inside parts. In summary, the weight of each collection R_j is greater than the weight of A_v^j. As $j \in [d+1]$, a heaviest collection among all R_js is a $d+1$-approximation of A_v.

Computing a $O(d^2)$-approx for A_r. *Construction.* For each $e = P_iP_j \in R(G/\mathcal{P})$, we compute an optimal solution for Max Leaves Induced Star Forest in $G[p(e)] = G[P_i \cup P_j]$ among those that are included in Y and have only edges with one endpoint in P_i and the other endpoint in P_j. Said differently, we determine a maximum-weighted collection of induced stars in $G[p(e)]$ over Y with a root on one side (for example, P_i) and all leaves on the other side (P_j). This costs at most $2^{O_d(\sqrt{n})}$ by trying out all vertex subsets, since $|P_i \cup P_i| \leqslant 2d\sqrt{n}$. The set of vertices of the solution returned on $G[p(e)]$ is denoted by $B_e \subseteq p(e)$.

For each $h \in [z]$, let H'_h be the trigraph $(G/\mathcal{P})[\{P_i : P_i \text{ is incident to an edge } e \in E_h\}]$. The red edges of H'_h form an induced matching on the red graph of H'_h as they are at distance 2 in $G/\mathcal{P}$. We associate with any edge $e \in E_h$ the edge weight of B_e. Then, we turn the red edges of H'_h in black: let H''_h be the graph obtained. We solve Max Subset Induced Matching on H''_h by restricting it to edges of E_h (which plays the role of Y): this is achieved in $2^{O(\sqrt{n})}$ as $|V(H''_h)| \leqslant \sqrt{n}$. Let I''_h be a maximum-weighted induced matching obtained. For each $h \in [z]$, we obtain the union R_h of all B_e, $e \in I''_h$: $R_h = \bigcup_{e \in I''_h} B_e$. We return an R_h which maximizes the total edge weight, among all $h \in [z]$.

Approximation ratio. Let A_r^h be the subset of A_r made up of edges being part of red edges E_h in $G/\mathcal{P}$, for $h \in [z]$. As the edges of E_h form an induced matching in $\mathcal{R}(G/\mathcal{P})$, the union of solutions of Max Leaves Induced Star Forest over graphs $G[p(e)]$ with $e \in E_h$ can only be connected through black edges of $G/\mathcal{P}$. Furthermore, two collections of stars over $G[p(e)]$ and $G[p(f)]$ are necessarily not mutually induced if there is a black edge between an endpoint of e and an endpoint of f. Consequently, R_h gives a maximum-weighted collection of mutually induced stars over E_h and its weight is at least the weight of A_r^h. The maximum-weighted collection over all R_h gives a z-approximation, as $h \in [z]$.

Computing a $2d+1$-approx for A_b. *Construction.* For each part P_i, we solve Weighted Max Independent Set on $G[P_i]$ with weight function w_V. Let $I(P_i)$ be the independent set returned and $w(P_i)$ its weight. We focus now on graph $G' = (V(G/\mathcal{P}), E(G/\mathcal{P}))$, made up of the black edges of $G/\mathcal{P}$, and solve Max Leaves Induced Star Forest on it with weights $w(P_i)$. As $|V(G')| \leqslant \sqrt{n}$, this is achieved in $2^{O(\sqrt{n})}$.

Let $(B_h)_{h\in[k]}$ be the collection of stars returned, $B_h = \{R_h, S_h^1, \ldots, S_h^{L_h}\}$ and $B \in E(G')$ be the set of edges belonging to this collection. Based on the bounded maximum red degree of $G/\mathcal{P}$, we determine a $O(d)$-partition of the edges of B, in order to produce collections of mutually induced stars. Let H^* be the graph where each edge e in the collection $(B_h)_{h\in[k]}$ is represented with a vertex and two of them e, f are adjacent if and only if there is a red edge in $G/\mathcal{P}$ connecting an endpoint of e with an endpoint of f. This graph has degree at most $2d$, so it can be $2d+1$-colored: let $T_1, \ldots, T_{2d+1}$ be the corresponding color classes. Any set of edges T_j gives us a collection of mutually induced stars on trigraph $G/\mathcal{P}$, in the sense that there is neither a black nor a red edge between two stars.

We fix some color class: say T_1 w.l.o.g. Let (B_h^*) be the collection of stars produced by T_1, where $B_h^* = \{R_h^*, S_h^{1,*}, \ldots, S_h^{L_h^*,*}\}$. For the root $R_h^* = P_i$ of each star B_h^*, we select an arbitrary vertex $r_h \in P_i$. Let $(B_h^**)_{h\in[k]}$ be the following collection of stars (which are mutually induced) on G: $B_h^{**} = \{r_h\} \cup \bigcup_{\ell=1}^{L_h^*} I(S_h^{\ell,*})$. In brief, the collection $(B_h^{**})_{h\in[k]}$ is made up of an arbitrary vertex of each root of stars B_h^* and a maximum-weighted

independent set of each leaf of B_h^*. Remember that we computed this collection of stars for T_1: we return a maximum-weighted collection $(B_h^{**})_{h\in[k]}$ among all the ones determined for T_j, $j \in [2d+1]$.

Approximation ratio. Any collection B_b with stars belonging only to black edges of $G/\mathcal{P}$ reveals a collection of stars on the quotient graph. Concretely, two black edges of $G/\mathcal{P}$ containing each a branch of B_b must be either non-adjacent or form an induced 3-vertex path on $G' = (V(G/\mathcal{P}), E(G/\mathcal{P}))$. Conversely, considering a collection B^* of mutually induced stars of G' and, for each $e \in B^*$, a collection B_e^* of mutually induced stars on $G[p(e)]$ produces a global collection of stars of G: then, we can partition its edges into $2d+1$ parts (as with $T_1, \ldots, T_{2d+1}$) such that each part contains mutually induced stars. As the collection B computed above provides us with a heaviest collection of G', a maximum-weighted B_h^{**} over all T_j is a $2d+1$-approximation for B, whose weight is at least the weight of A_b.

Conclusion of the proof. We finally output a heaviest collection of mutually induced stars among the three approximating respectively A_v, A_r, and A_b. The overall running time is in $2^{O_d(\sqrt{n})}$. An upper bound for the approximation ratio of this algorithm is $3z = O(d^2)$. □

As for the other problems treated in this article, we apply to Max Leaves Induced Star Forest the time-approximation trade-off proposed in lemma 79.

Theorem 93. *Max Leaves Induced Star Forest on an n-vertex graph G, weight function w_V, with prescribed set $Y \subseteq E(G)$, and given with a d'-sequence, satisfies the assumptions of lemma 79. In particular, this problem admits*

- *a $(d+1)^{2^q-1}$-approximation in time $2^{O_{d,q}(n^{2^{-q}})}$, for every integer $q \geqslant 0$,*
- *an n^{ε}-approximation in polynomial-time $O_{d,\varepsilon}(1)\log^{O_d(1)} n \cdot n^{O(1)}$, for any $\varepsilon > 0$, and*
- *a $\log n$-approximation in time $2^{O_d(n^{\frac{1}{\log\log n}})}$,*

with $d := c_{2d'+2} \cdot 2^{4c_{2d'+2}+4}$.

Proof. The exhaustive algorithm (trying out all vertex subsets and checking whether they induce a collection of mutually induced stars in Y) solves Max Leaves Induced Star Forest in time $2^{O(n)}$. Thus we show Max Leaves Induced Star Forest satisfies the second item of lemma 79. We set $c_2 \geqslant 1$ as the required exponent to turn a d'-sequence into a neatly divided matrix of $\mathcal{M}_{n,2d'+2}$ conform to G, and compute the various needed colorings, the appropriate $\frac{1}{2} < c_1 < 1$, and $2 < c_3 < 3$, and $f(d) = 2d \geqslant 1$.

Approximating A_v. The algorithm makes $\lfloor\sqrt{n}\rfloor$ recursive calls to solve Max Leaves Induced Star Forest on parts P_i. Furthermore, $d+1$ calls to WMIS are needed on induced subgraphs of $G/\mathcal{P}$. All of these induced subgraphs are on at most $d\sqrt{n}$ vertices.

Approximating A_r. The algorithm makes at most $\frac{\sqrt{n}d}{2}$ recursive calls (one call per red edge of $G/\mathcal{P}$) on induced subgraphs of G with at most $2d\sqrt{n}$ vertices, followed by at most $2(d-1)d+1$ calls of Max Subset Induced Matching on full cleanups of induced subtrigraphs of $G/\mathcal{P}$ with at most $\sqrt{n}$ vertices.

Approximating A_b. The algorithm makes $\lfloor\sqrt{n}\rfloor$ calls to solve WMIS on parts P_i and one recursive call on a full cleanup of $G/\mathcal{P}$ on $\lfloor\sqrt{n}\rfloor$ vertices.

In summary, we make $O_d(\sqrt{n})$ recursive calls or calls to problems WMIS and Max Subset Induced Matching (which already satisfy lemma 79 with better constants) on induced subgraphs of G or full cleanups of (the whole) $G/\mathcal{P}$, each on $O_d(\sqrt{n})$ vertices. Hence, by lemma 75, the induction applies.

Getting r-approximations on every subcall allows us to output a global $3(2(d-1)d+1)r^2$-approximation for Max Leaves Induced Star Forest:

- collection A_v is approximated with ratio $(d+1)r^2$
- collection A_r is approximated with ratio $(2(d-1)d+1)r^2$
- collection A_b is approximated with ratio $(2d+1)r^2$.

The extra factor 3 comes from the fact that we output the heaviest of these three solutions. □

Max Edge Induced Star Forest is a particular case of Max Leaves Induced Star Forest with $w_V(u) = 1$ for every vertex $u \in V(G)$. Furthermore, a solution of Max Edge Induced Star Forest is a 3-approximation of a solution of Max Edge Induced Forest. These observations together with theorem 93 allow us to state the following result.

Corollary 94. *Max Edge Induced Star Forest and Max Edge Induced Forest on an n-vertex graph G, with prescribed set $Y \subseteq E(G)$, and given with a d'-sequence, admit*

- *an n^ε-approximation in polynomial-time $O_{d,\varepsilon}(1)\log^{O_d(1)} n \cdot n^{O(1)}$, for any $\varepsilon > 0$, and*
- *a $\log n$-approximation in time $2^{O_d(n^{\frac{1}{\log\log n}})}$,*

with $d := c_{2d'+2} \cdot 2^{4c_{2d'+2}+4}$.

3.6 Limits

We now discuss the limits of our framework. We give some examples of problems that are unlikely to have an n^ε-approximation algorithm on graphs of bounded twin-width. The first such problem is Min Independent Dominating Set, where one seeks a minimum-cardinality set which is both an independent set and a dominating set. In general n-vertex graphs, this problem cannot be $n^{1-\varepsilon}$-approximated in polynomial time unless P=NP [68], and cannot be r-approximated in time $2^{o(n/r)}$ for any $r = r(n)$, unless the ETH fails [75].

We show that Min Independent Dominating Set has the same polytime inapproximability in graphs of bounded twin-width.

Theorem 95. *For every $\varepsilon > 0$, Min Independent Dominating Set cannot be $n^{1-\varepsilon}$-approximated in polynomial time on n-vertex graphs of twin-width at most 9 given with a 9-sequence, unless P=NP.*

Proof. We perform the classic reduction of Halldórsson from SAT [68], but from Planar 3-SAT where each literal has at most two occurrences, which remains NP-complete [76]. More precisely we add a triangle d_i, t_i, f_i for each variable x_i (with $i \in [N]$), and an independent set I_j of size r for each 3-clause C_j (with $j \in [M]$). We link t_i to all the vertices of I_j whenever x_i appears positively in C_j, and we link f_i to all the vertices of I_j whenever x_i appears negatively in C_j. This defines a graph G with $n = 3N + rM$ vertices.

It can be observed that if the Planar 3-SAT instance is satisfiable, then there is an independent dominating set of size N, whereas if the formula is unsatisfiable then any independent dominating set has size at least r. Setting $r := N^{\frac{2-\varepsilon}{\varepsilon}}$, the gap between positive and negative instances is $\Theta_\varepsilon(1)n^{1-\varepsilon}$, while preserving the fact that the reduction is polynomial.

Let us now argue that G has twin-width at most 9, and that a 9-sequence of it can be computed in polynomial time. We can first contract each I_j into a single vertex without creating a red edge. Next we can contract every triangle d_i, t_i, f_i into a single vertex of red degree at most 4. At this point, the current trigraph is a planar graph of maximum degree at most 4. It was observed in [29] that planar trigraphs with maximum (total) degree at

most 9 have twin-width at most 9. This is because any planar graph has a pair of vertices on the same face with at most 9 neighbors (outside of themselves) combined [77]. Hence we get a 9-sequence for G that can be computed in polynomial time. (One could also use the more complicated linear algorithm to get a *8*-sequence for *any* planar graph [9].) □

Another very inapproximable is Longest Induced Path, which also does not admit a polytime $n^{1-\varepsilon}$-approximation algorithm unless P=NP [70], and cannot be r-approximated in time $2^{o(n/r)}$ for any $r = o(n)$, unless the ETH fails [75]. The non-induced version, the Longest Path problem, has a notoriously big gap between the best known approximation algorithm whose factor is $n/\exp(\Omega(\sqrt{\log n}))$ [69], and the sharpest conditional lower bound which states that, for any $\varepsilon > 0$, a $2^{\log^{1-\varepsilon} n}$-approximation would imply that NP $\subseteq$ QP [78].

Despite being an open question for decades the existence or conditional impossibility of an approximation algorithm for Longest Path with approximation factor, say, $\sqrt{n}$ has not been settled. Nor do we know whether an n^{ε}-approximation for any $\varepsilon > 0$ is possible. We now show that using our framework to obtain an n^{ε}-approximation for Longest Induced Path of Longest Path in graphs of bounded twin-width is unlikely to work, in the sense that it would immediately yield such an approximation factor for Longest Path in general graphs.

Theorem 96. *For any $r = \omega(1)$, an r-approximation for Longest Induced Path or Longest Path on graphs of twin-width at most 4 given with a 4-sequence would imply a $(1 + o(1))r$-approximation for Longest Path in general graphs.*

Proof. It was shown in [28] that any graph obtained by subdividing every edge of an n-vertex graph at least $2 \log n$ has twin-width at most 4. Besides, a 4-sequence can then be computed in polynomial time.

Let G be any graph with minimum degree at least 2 (note that this restriction does not make Longest Path easier to approximate), and G' be obtained from G by subdividing each of its edges $2\lceil \log n \rceil$ times, and let $s := 2\lceil \log n \rceil + 1$. Let us observe that G has a path of length ℓ if and only if G' has a path of length $(\ell + 2)s - 2$ if and only if G' has an induced path of length $(\ell + 2)s - 4$. Hence a polytime r-approximation for Longest Induced Path or Longest Path in graphs of bounded twin-width given a 4-sequence would translate into a $(1+o(1))r$-approximation for Longest Path in general graphs. □

We can use theorem 96 to get a similar weak obstruction to an n^{ε}-approximation for Mutually Induced $\mathcal{H}$-packing in graphs of bounded twin-width, for some infinite family of connected graphs $\mathcal{H}$. Recall that by lemma 91 such an approximation algorithm does exist when $\mathcal{H}$ is a *finite* collection of connected graphs.

Setting $\mathcal{H}$ to be the set of all paths does not serve that purpose, since one can then use the approximation algorithm for Max Induced Matching. Nevertheless this almost works. We just need to *decorate* the endpoints of the paths. For every positive integer n, let D_n be the *decorated path* of length n, obtained from the n-vertex path P_n by adding for each endpoint u two adjacent vertices u', u'' both adjacent to u. Informally, D_n is a path terminated by a triangle at each end.

Theorem 97. *Let $\mathcal{H} := \{D_n : n \in \mathbb{N}^+\}$ be the family of all decorated paths. If for every $\varepsilon > 0$, Mutually Induced $\mathcal{H}$-packing admits an n^{ε} on n-vertex graphs of bounded twin-width given with a 4-sequence, then so does Longest Path on general graphs.*

Proof. Let G be any graph. For every pair $u \neq v \in V(G)$, define G_{uv} as the graph obtained from G by subdividing all its edges $2\lceil \log(n+2) \rceil$ times, and adding two adjacent

vertices u', u'' both adjacent to u, and two adjacent vertices v', v'' both adjacent to v. Since there are only two triangles in G_{uv}, only one graph of $\mathcal{H}$ can be present in a (mutually induced) packing. Thus MUTUALLY INDUCED $\mathcal{H}$-PACKING is now equivalent to finding a longest path between u and v. An n^{ε}-approximation algorithm for this problem would, by theorem 96, give a similar approximation algorithm for LONGEST PATH in general graphs.

Despite u', u'', v', v'', G_{uv} still admits a 4-sequence. For instance, first contract u' and u'', and contract v' and v''; this does not create red edges, and has the same effect as deleting u'' and v''. The obtained graph is an induced subgraph of a $2\lceil\log(n+2)\rceil$-subdivision (of a graph on at most $n+2$ vertices). Hence it admits a polytime computable 4-sequence [28]. □

Conclusion

In Chapter 1, we presented a construction whose twin-width is exponential in treewidth. An interesting point about this construction is the proof of the lower bound on twin-width. It is quite hard to lower bound twin-width because we lack a general construction showing that a graph has twin-width at least d for a given d, like brambles for treewidth:

▷ Question 5. For a given d, is there a construction such that the graph has twin-width at least d if and only if the graph has the construction?

It is known from [5] that there exists a class of graphs of degree at most 3 with unbounded twin-width, but no explicit construction is known. Perhaps the construction presented in Section 1.2 can be adapted to find such a class.

▷ Question 6. Can we give an explicit construction of a class of graphs with degree at most 3 and unbounded twin-width?

For the relation between the mixed minor parameter and twin-width, there is a gap between our lower bound and the upper bound given by Theorem 14 of [4].

▷ Question 7. Is the double exponential bound on twin-width in function of the mixed-minor parameter optimal?

In Chapter 2, we showed that deciding if twin-width is at most 4 is NP-hard. Even if it is hard, computing twin-width is still a crucial question. Our result only rules out anything better than a 5/4 approximation algorithm.

▷ Question 8. Is there a polynomial-time $f(n)$-approximation algorithm for computing twin-width with $f(n) = o(n)$?

We can also ask whether the hardness result holds if we restrict ourselves to simpler classes of graphs. We believe that this is the case for graphs of bounded treewidth.

Conjecture 98. *Deciding, given an integer k, if a graph of bounded treewidth has twin-width at most k is NP-hard.*

In Chapter 3, we give several optimization problems that are easier to approximate on graphs of bounded twin-width. We also give some problems that are still hard to approximate.

For the Maximum Independent Set problem, there is no known lower bound on the approximation ratio of a polynomial-time algorithm on graphs of bounded twin-width.

▷ Question 9. Is it possible to show a hardness result matching our $O(n^\epsilon)$-approximation algorithm?

If not, can a better approximation algorithm be found? From [14], we know that a constant ratio approximation algorithm implies that there is a PTAS.

▷ Question 10. Is there a Polynomial Time Approximation Scheme for Maximum Independent Set on graphs of bounded twin-width?

To give an example of a problem for which we may be closer to finding such an algorithm, there is Minimum Dominating Set. From [14], we already know that there is an approximation algorithm with a constant ratio.

▷ Question 11. Is there a PTAS for Minimum Dominating Set on graphs of bounded twin-width?

Bibliography

[1] Bruno Courcelle. "The monadic second-order logic of graphs. I. Recognizable sets of finite graphs". In: *Information and Computation* 85.1 (1990), pp. 12–75. ISSN: 0890-5401. DOI: https://doi.org/10.1016/0890-5401(90)90043-H. URL: http://www.sciencedirect.com/science/article/pii/089054019090043H.

[2] Hans L. Bodlaender. "A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth". In: *SIAM J. Comput.* 25.6 (1996), pp. 1305–1317. URL: https://doi.org/10.1137/S0097539793251219.

[3] Sylvain Guillemot and Dániel Marx. "Finding small patterns in permutations in linear time". In: *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*. 2014, pp. 82–101. DOI: 10.1137/1.9781611973402.7. URL: https://doi.org/10.1137/1.9781611973402.7.

[4] Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. "Twin-width I: Tractable FO Model Checking". In: *J. ACM* 69.1 (2022), 3:1–3:46. URL: https://doi.org/10.1145/3486655.

[5] Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. "Twin-width II: small classes". In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2021, pp. 1977–1996. DOI: 10.1137/1.9781611976465.118.

[6] Édouard Bonnet, O-joung Kwon, and David R. Wood. "Reduced bandwidth: a qualitative strengthening of twin-width in minor-closed classes (and beyond)". In: *CoRR* abs/2202.11858 (2022). URL: https://arxiv.org/abs/2202.11858.

[7] Édouard Bonnet and Julien Duron. "Stretch-Width". In: *18th International Symposium on Parameterized and Exact Computation, IPEC 2023, September 6-8, 2023, Amsterdam, The Netherlands*. Ed. by Neeldhara Misra and Magnus Wahlström. Vol. 285. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, 8:1–8:15. URL: https://doi.org/10.4230/LIPIcs.IPEC.2023.8.

[8] Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Pierre Simon, Stéphan Thomassé, and Szymon Toruńczyk. "Twin-width IV: ordered graphs and matrices". In: *CoRR* abs/2102.03117 (2021). arXiv: 2102.03117. URL: https://arxiv.org/abs/2102.03117.

[9] Petr Hlinený and Jan Jedelský. "Twin-Width of Planar Graphs Is at Most 8, and at Most 6 When Bipartite Planar". In: *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*. Ed. by Kousha Etessami, Uriel Feige, and Gabriele Puppis. Vol. 261. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, 75:1–75:18. URL: https://doi.org/10.4230/LIPIcs.ICALP.2023.75.

[10] Daniel Král and Ander Lamaison. "Planar graph with twin-width seven". In: *CoRR* abs/2209.11537 (2022). URL: https://doi.org/10.48550/arXiv.2209.11537.

[11] Hugo Jacob and Marcin Pilipczuk. "Bounding twin-width for bounded-treewidth graphs, planar graphs, and bipartite graphs". In: *CoRR* abs/2201.09749 (2022). URL: https://arxiv.org/abs/2201.09749.

[12] Petr Hliněný. *Twin-width of Planar Graphs is at most 9*. 2022. DOI: 10.48550/ARXIV.2205.05378. URL: https://arxiv.org/abs/2205.05378.

[13] Édouard Bonnet, Dibyayan Chakraborty, Eun Jung Kim, Noleen Köhler, Raul Lopes, and Stéphan Thomassé. "Twin-width VIII: delineation and win-wins". In: *CoRR* abs/2204.00722 (2022). URL: https://doi.org/10.48550/arXiv.2204.00722.

[14] Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. "Twin-width III: Max Independent Set, Min Dominating Set, and Coloring". In: *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*. Ed. by Nikhil Bansal, Emanuela Merelli, and James Worrell. Vol. 198. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 35:1–35:20. URL: https://doi.org/10.4230/LIPIcs.ICALP.2021.35.

[15] Michal Pilipczuk and Marek Sokolowski. "Graphs of bounded twin-width are quasi-polynomially χ-bounded". In: *CoRR* abs/2202.07608 (2022). URL: https://arxiv.org/abs/2202.07608.

[16] William Pettersson and John Sylvester. "Bounds on the Twin-Width of Product Graphs". In: *CoRR* abs/2202.11556 (2022). URL: https://arxiv.org/abs/2202.11556.

[17] Édouard Bonnet, Eun Jung Kim, Amadeus Reinald, Stéphan Thomassé, and Rémi Watrigant. "Twin-Width and Polynomial Kernels". In: *16th International Symposium on Parameterized and Exact Computation, IPEC 2021, September 8-10, 2021, Lisbon, Portugal*. Ed. by Petr A. Golovach and Meirav Zehavi. Vol. 214. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 10:1–10:16. URL: https://doi.org/10.4230/LIPIcs.IPEC.2021.10.

[18] Wojciech Przybyszewski. *VC-density and abstract cell decomposition for edge relation in graphs of bounded twin-width*. 2022. URL: https://arxiv.org/abs/2202.04006.

[19] Stefan Kratsch, Florian Nelles, and Alexandre Simon. "On Triangle Counting Parameterized by Twin-Width". In: *CoRR* abs/2202.06708 (2022). URL: https://arxiv.org/abs/2202.06708.

[20] Michal Pilipczuk, Marek Sokolowski, and Anna Zych-Pawlewicz. "Compact Representation for Matrices of Bounded Twin-Width". In: *39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022, March 15-18, 2022, Marseille, France (Virtual Conference)*. Ed. by Petra Berenbrink and Benjamin Monmege. Vol. 219. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 52:1–52:14. URL: https://doi.org/10.4230/LIPIcs.STACS.2022.52.

[21] Jakub Balabán and Petr Hlinený. "Twin-Width Is Linear in the Poset Width". In: *16th International Symposium on Parameterized and Exact Computation, IPEC 2021, September 8-10, 2021, Lisbon, Portugal*. Ed. by Petr A. Golovach and Meirav Zehavi. Vol. 214. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 6:1–6:13. URL: https://doi.org/10.4230/LIPIcs.IPEC.2021.6.

[22] Jakub Balabán, Petr Hlinený, and Jan Jedelský. "Twin-width and Transductions of Proper k-Mixed-Thin Graphs". In: *CoRR* abs/2202.12536 (2022). URL: https://arxiv.org/abs/2202.12536.

[23] André Schidler and Stefan Szeider. "A SAT Approach to Twin-Width". In: *CoRR (accepted at ALENEX '22)* abs/2110.06146 (2021). URL: https://arxiv.org/abs/2110.06146.

[24] Daniel Král, Kristýna Pekárková, and Kenny Storgel. "Twin-width of graphs on surfaces". In: *CoRR* abs/2307.05811 (2023). URL: https://doi.org/10.48550/arXiv.2307.05811.

[25] Jungho Ahn, Debsoumya Chakraborti, Kevin Hendrey, and Sang-il Oum. "Twin-width of subdivisions of multigraphs". In: *CoRR* abs/2306.05334 (2023). URL: https://doi.org/10.48550/arXiv.2306.05334.

[26] Jungho Ahn, Kevin Hendrey, Donggyu Kim, and Sang-il Oum. "Bounds for the Twin-width of Graphs". In: *CoRR* abs/2110.03957 (2021). URL: https://arxiv.org/abs/2110.03957.

[27] Yonatan Bilu and Nathan Linial. "Lifts, Discrepancy and Nearly Optimal Spectral Gap*". In: *Combinatorica* 26.5 (2006), pp. 495–519. DOI: 10.1007/s00493-006-0029-7. URL: https://doi.org/10.1007/s00493-006-0029-7.

[28] Pierre Bergé, Édouard Bonnet, and Hugues Déprés. "Deciding twin-width at most 4 is NP-complete". In: *CoRR* abs/2112.08953 (2021). URL: https://arxiv.org/abs/2112.08953.

[29] Édouard Bonnet, Eun Jung Kim, Amadeus Reinald, and Stéphan Thomassé. "Twin-width VI: the lens of contraction sequences". In: *CoRR, accepted at SODA 2022* abs/2111.00282 (2021). URL: https://arxiv.org/abs/2111.00282.

[30] Jörg Flum and Martin Grohe. "Fixed-Parameter Tractability, Definability, and Model-Checking". In: *SIAM J. Comput.* 31.1 (2001), pp. 113–145. URL: https://doi.org/10.1137/S0097539799360768.

[31] Jakub Gajarský, Petr Hlinený, Jan Obdrzálek, and Sebastian Ordyniak. "Faster Existential FO Model Checking on Posets". In: *Logical Methods in Computer Science* 11.4 (2015). DOI: 10.2168/LMCS-11(4:8)2015. URL: https://doi.org/10.2168/LMCS-11(4:8)2015.

[32] Robert Ganian, Petr Hlinený, Daniel Král, Jan Obdrzálek, Jarett Schwartz, and Jakub Teska. "FO Model Checking of Interval Graphs". In: *Logical Methods in Computer Science* 11.4 (2015). DOI: 10.2168/LMCS-11(4:11)2015. URL: https://doi.org/10.2168/LMCS-11(4:11)2015.

[33] Jakub Gajarský, Petr Hlinený, Daniel Lokshtanov, Jan Obdrzálek, Sebastian Ordyniak, M. S. Ramanujan, and Saket Saurabh. "FO Model Checking on Posets of Bounded Width". In: *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*. 2015, pp. 963–974. DOI: 10.1109/FOCS.2015.63. URL: https://doi.org/10.1109/FOCS.2015.63.

[34] Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. "Deciding First-Order Properties of Nowhere Dense Graphs". In: *J. ACM* 64.3 (2017), 17:1–17:32. DOI: 10.1145/3051095. URL: https://doi.org/10.1145/3051095.

[35] Zdenek Dvorák, Daniel Král, and Robin Thomas. "Testing first-order properties for subclasses of sparse graphs". In: *J. ACM* 60.5 (2013), 36:1–36:24. DOI: 10.1145/2499483. URL: https://doi.org/10.1145/2499483.

[36] Stephan Kreutzer and Anuj Dawar. "Parameterized Complexity of First-Order Logic". In: *Electronic Colloquium on Computational Complexity (ECCC)* 16 (2009), p. 131. URL: http://eccc.hpi-web.de/report/2009/131.

[37] Uriel Feige, MohammadTaghi Hajiaghayi, and James R. Lee. "Improved Approximation Algorithms for Minimum Weight Vertex Separators". In: *SIAM J. Comput.* 38.2 (2008), pp. 629–657. DOI: 10.1137/05064299X. URL: https://doi.org/10.1137/05064299X.

[38] Tuukka Korhonen. "Single-Exponential Time 2-Approximation Algorithm for Treewidth". In: *CoRR* abs/2104.07463 (2021). URL: https://arxiv.org/abs/2104.07463.

[39] Sang-il Oum. "Approximating rank-width and clique-width quickly". In: *ACM Trans. Algorithms* 5.1 (2008), 10:1–10:20. URL: https://doi.org/10.1145/1435375.1435385.

[40] Stefan Arnborg, Derek G Corneil, and Andrzej Proskurowski. "Complexity of finding embeddings in a k-tree". In: *SIAM Journal on Algebraic Discrete Methods* 8.2 (1987), pp. 277–284.

[41] Tatsuo Ohtsuki, Hajimu Mori, Ernest S. Kuh, Toshinobu Kashiwabara, and Toshio Fujisawa. "One-dimensional logic gate assignment and interval graphs". In: *The IEEE Computer Society's Third International Computer Software and Applications Conference, COMPSAC 1979, 6-8 November, 1979, Chicago, Illinois, USA*. IEEE, 1979, pp. 101–106. DOI: 10.1109/CMPSAC.1979.762474. URL: https://doi.org/10.1109/CMPSAC.1979.762474.

[42] Toshinobu Kashiwabara. "NP-completeness of the problem of finding a minimal-clique number interval graph containing a given graph as a subgraph". In: *Proc. 1979 Int. Symp. Circuit Syst.* 1979, pp. 657–660.

[43] Thomas Lengauer. "Black-White Pebbles and Graph Separation". In: *Acta Informatica* 16 (1981), pp. 465–475. DOI: 10.1007/BF00264496. URL: https://doi.org/10.1007/BF00264496.

[44] Michael R. Fellows, Frances A. Rosamond, Udi Rotics, and Stefan Szeider. "Clique-Width is NP-Complete". In: *SIAM J. Discret. Math.* 23.2 (2009), pp. 909–939. DOI: 10.1137/070687256. URL: https://doi.org/10.1137/070687256.

[45] Petr Hlinený and Sang-il Oum. "Finding Branch-Decompositions and Rank-Decompositions". In: *SIAM J. Comput.* 38.3 (2008), pp. 1012–1032. URL: https://doi.org/10.1137/070685920.

[46] Sigve Hortemo Sæther and Martin Vatshelle. "Hardness of computing width parameters based on branch decompositions over the vertex set". In: *Theor. Comput. Sci.* 615 (2016), pp. 120–125. DOI: 10.1016/j.tcs.2015.11.039. URL: https://doi.org/10.1016/j.tcs.2015.11.039.

[47] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. ISBN: 0-7167-1044-7.

[48] Neil Robertson and Paul D. Seymour. "Graph Minors. XIII. The Disjoint Paths Problem". In: *J. Comb. Theory, Ser. B* 63.1 (1995), pp. 65–110. URL: https://doi.org/10.1006/jctb.1995.1006.

[49] James B. Saxe. "Dynamic-Programming Algorithms for Recognizing Small-Bandwidth Graphs in Polynomial Time". In: *SIAM J. Algebraic Discret. Methods* 1.4 (1980), pp. 363–369. URL: https://doi.org/10.1137/0601042.

[50] Markus Sortland Dregi and Daniel Lokshtanov. "Parameterized Complexity of Bandwidth on Trees". In: *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*. Ed. by Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias. Vol. 8572. Lecture Notes in Computer Science. Springer, 2014, pp. 405–416. URL: https://doi.org/10.1007/978-3-662-43948-7%5C_34.

[51] Derek G. Corneil, Michel Habib, Jean-Marc Lanlignel, Bruce A. Reed, and Udi Rotics. "Polynomial-time recognition of clique-width $\leqslant 3$ graphs". In: *Discret. Appl. Math.* 160.6 (2012), pp. 834–865. DOI: 10.1016/j.dam.2011.03.020. URL: https://doi.org/10.1016/j.dam.2011.03.020.

[52] Michel Habib and Christophe Paul. "A simple linear time algorithm for cograph recognition". In: *Discret. Appl. Math.* 145.2 (2005), pp. 183–197. DOI: 10.1016/j.dam.2004.01.011. URL: https://doi.org/10.1016/j.dam.2004.01.011.

[53] Marek Cygan, Fedor V. Fomin, Alexander Golovnev, Alexander S. Kulikov, Ivan Mihajlin, Jakub Pachocki, and Arkadiusz Socala. "Tight Lower Bounds on Graph Embedding Problems". In: *J. ACM* 64.3 (2017), 18:1–18:22. URL: https://doi.org/10.1145/3051094.

[54] Fedor V. Fomin, Daniel Lokshtanov, Ivan Mihajlin, Saket Saurabh, and Meirav Zehavi. "Computation of Hadwiger Number and Related Contraction Problems: Tight Lower Bounds". In: *ACM Trans. Comput. Theory* 13.2 (2021), 10:1–10:25. DOI: 10.1145/3448639. URL: https://doi.org/10.1145/3448639.

[55] Russell Impagliazzo and Ramamohan Paturi. "On the Complexity of k-SAT". In: *J. Comput. Syst. Sci.* 62.2 (2001), pp. 367–375. DOI: 10.1006/jcss.2000.1727. URL: https://doi.org/10.1006/jcss.2000.1727.

[56] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. "Which Problems Have Strongly Exponential Complexity?" In: *J. Comput. Syst. Sci.* 63.4 (2001), pp. 512–530. DOI: 10.1006/jcss.2001.1774. URL: https://doi.org/10.1006/jcss.2001.1774.

[57] Craig A. Tovey. "A simplified NP-complete satisfiability problem". In: *Discret. Appl. Math.* 8.1 (1984), pp. 85–89. URL: https://doi.org/10.1016/0166-218X(84)90081-7.

[58] Johan Håstad. "Clique is Hard to Approximate Within $n^{1-\epsilon}$". In: *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*. 1996, pp. 627–636. DOI: 10.1109/SFCS.1996.548522. URL: https://doi.org/10.1109/SFCS.1996.548522.

[59] David Zuckerman. "Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number". In: *Theory of Computing* 3.1 (2007), pp. 103–128.

[60] Parinya Chalermsook, Bundit Laekhanukit, and Danupon Nanongkai. "Graph Products Revisited: Tight Approximation Hardness of Induced Matching, Poset Dimension and More". In: *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*. Ed. by Sanjeev Khanna. SIAM, 2013, pp. 1557–1576. DOI: 10.1137/1.9781611973105.112. URL: https://doi.org/10.1137/1.9781611973105.112.

[61] Jakub Gajarský, Michal Pilipczuk, Wojciech Przybyszewski, and Szymon Torunczyk. "Twin-Width and Types". In: *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*. Ed. by Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff. Vol. 229. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 123:1–123:21. DOI: 10.4230/LIPIcs.ICALP.2022.123. URL: https://doi.org/10.4230/LIPIcs.ICALP.2022.123.

[62] Irit Dinur and David Steurer. "Analytical approach to parallel repetition". In: *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*. Ed. by David B. Shmoys. ACM, 2014, pp. 624–633. DOI: 10.1145/2591796.2591884. URL: https://doi.org/10.1145/2591796.2591884.

[63] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[64] Marek Cygan, Lukasz Kowalik, Marcin Pilipczuk, and Mateusz Wykurz. "Exponential-Time Approximation of Hard Problems". In: *CoRR* abs/0810.4934 (2008). arXiv: 0810.4934. URL: http://arxiv.org/abs/0810.4934.

[65] Nicolas Bourgeois, Bruno Escoffier, and Vangelis Th. Paschos. "Approximation of max independent set, min vertex cover and related problems by moderately exponential algorithms". In: *Discret. Appl. Math.* 159.17 (2011), pp. 1954–1970. DOI: 10.1016/j.dam.2011.07.009. URL: https://doi.org/10.1016/j.dam.2011.07.009.

[66] Nikhil Bansal, Parinya Chalermsook, Bundit Laekhanukit, Danupon Nanongkai, and Jesper Nederlof. "New Tools and Connections for Exponential-Time Approximation". In: *Algorithmica* 81.10 (2019), pp. 3993–4009. DOI: 10.1007/s00453-018-0512-8. URL: https://doi.org/10.1007/s00453-018-0512-8.

[67] Parinya Chalermsook, Bundit Laekhanukit, and Danupon Nanongkai. "Independent Set, Induced Matching, and Pricing: Connections and Tight (Subexponential Time) Approximation Hardnesses". In: *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*. 2013, pp. 370–379. DOI: 10.1109/FOCS.2013.47. URL: https://doi.org/10.1109/FOCS.2013.47.

[68] Magnús M. Halldórsson. "Approximating the Minimum Maximal Independence Number". In: *Inf. Process. Lett.* 46.4 (1993), pp. 169–172. DOI: 10.1016/0020-0190(93)90022-2. URL: https://doi.org/10.1016/0020-0190(93)90022-2.

[69] Harold N. Gabow and Shuxin Nie. "Finding a long directed cycle". In: *ACM Trans. Algorithms* 4.1 (2008), 7:1–7:21. DOI: 10.1145/1328911.1328918. URL: https://doi.org/10.1145/1328911.1328918.

[70] Carsten Lund and Mihalis Yannakakis. "The Approximation of Maximum Subgraph Problems". In: *Automata, Languages and Programming, 20nd International Colloquium, ICALP93, Lund, Sweden, July 5-9, 1993, Proceedings*. Ed. by Andrzej Lingas, Rolf G. Karlsson, and Svante Carlsson. Vol. 700. Lecture Notes in Computer Science. Springer, 1993, pp. 40–51. DOI: 10.1007/3-540-56939-1_60. URL: https://doi.org/10.1007/3-540-56939-1%5C_60.

[71] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. "Approximating Clique is Almost NP-Complete (Preliminary Version)". In: *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*. IEEE Computer Society, 1991, pp. 2–12. DOI: 10.1109/SFCS.1991.185341. URL: https://doi.org/10.1109/SFCS.1991.185341.

[72] Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. "Twin-width II: small classes". In: *CoRR* abs/2006.09877 (2020, to appear at SODA 2021). arXiv: 2006.09877. URL: https://arxiv.org/abs/2006.09877.

[73] Noga Alon, Guoli Ding, Bogdan Oporowski, and Dirk Vertigan. "Partitioning into graphs with only small components". In: *J. Comb. Theory, Ser. B* 87.2 (2003), pp. 231–243. DOI: 10.1016/S0095-8956(02)00006-0.

[74] Jesper Nederlof. *Inclusion exclusion for hard problems*. Master thesis. 2008.

[75] Édouard Bonnet, Michael Lampis, and Vangelis Th. Paschos. "Time-approximation trade-offs for inapproximable problems". In: *J. Comput. Syst. Sci.* 92 (2018), pp. 171–180. DOI: 10.1016/j.jcss.2017.09.009. URL: https://doi.org/10.1016/j.jcss.2017.09.009.

[76] László Kozma. "Minimum Average Distance Triangulations". In: *Algorithms - ESA 2012 - 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings*. Ed. by Leah Epstein and Paolo Ferragina. Vol. 7501. Lecture Notes in Computer Science. Springer, 2012, pp. 695–706. DOI: 10.1007/978-3-642-33090-2_60. URL: https://doi.org/10.1007/978-3-642-33090-2%5C_60.

[77] Anton Kotzig. "Contribution to the theory of Eulerian polyhedra". In: *Mat. Cas. SAV (Math. Slovaca)* 5 (1955), pp. 111–113.

[78] David R. Karger, Rajeev Motwani, and G. D. S. Ramkumar. "On Approximating the Longest Path in a Graph". In: *Algorithmica* 18.1 (1997), pp. 82–98. DOI: 10.1007/BF02523689. URL: https://doi.org/10.1007/BF02523689.