Twin-width V: linear minors, modular counting, and matrix multiplication

Edouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Stéphan Thomassé

Twin-width V: linear minors, modular counting, and matrix multiplication

Édouard Bonnet
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France

Ugo Giocanti
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France

Patrice Ossona de Mendez
Centre d'Analyse et de Mathématique Sociales CNRS UMR 8557, France
and Computer Science Institute of Charles University (IUUK), Praha, Czech Republic

Stéphan Thomassé
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France

Abstract

We continue developing the theory around the twin-width of totally ordered binary structures (or equivalently, matrices over a finite alphabet), initiated in the previous paper of the series. We first introduce the notion of parity and linear minors of a matrix, which consists of iteratively replacing consecutive rows or consecutive columns with a linear combination of them. We show that a matrix class (i.e., a set of matrices closed under taking submatrices) has bounded twin-width if and only if its linear-minor closure does not contain all matrices. We observe that the fixed-parameter tractable (FPT) algorithm for first-order model checking on structures given with an $\mathcal{O}(1)$-sequence (certificate of bounded twin-width) and the fact that first-order transductions of bounded twin-width classes have bounded twin-width, both established in *Twin-width I*, extend to first-order logic with modular counting quantifiers. We make explicit a win-win argument obtained as a by-product of *Twin-width IV*, and somewhat similar to bidimensionality, that we call rank-bidimensionality. This generalizes the seminal work of Guillemot and Marx [SODA '14], which builds on the Marcus-Tardos theorem [JCTA '04]. It works on general matrices (not only on classes of bounded twin-width) and, for example, yields FPT algorithms deciding if a small matrix is a parity or a linear minor of another matrix given in input, or exactly computing the grid or mixed number of a given matrix (i.e., the maximum integer k such that the row set and the column set of the matrix can be partitioned into k intervals, with each of the k^2 defined cells containing a non-zero entry, or two distinct rows and two distinct columns, respectively).

Armed with the above-mentioned extension to modular counting, we show that the twin-width of the product of two conformal matrices A, B (i.e., whose dimensions are such that AB is defined) over a finite field is bounded by a function of the twin-width of A, of B, and of the size of the field. Furthermore, if A and B are $n \times n$ matrices of twin-width d over $\mathbb{F}_q$, we show that AB can be computed in time $\mathcal{O}_{d,q}(n^2 \log n)$.

We finally present an *ad hoc* algorithm to efficiently multiply two matrices of bounded twin-width, with a single-exponential dependence in the twin-width bound. More precisely, pipelined to observations and results of Pilipczuk et al. [STACS '22], we obtain the following. If the inputs are given in a compact tree-like form (witnessing twin-width at most d), called twin-decomposition of width d, then two $n \times n$ matrices A, B over $\mathbb{F}_2$ can be multiplied in time $4^{d+o(d)}n$, in the sense that a twin-decomposition of their product AB, with width $2^{d+o(d)}$, is output within that time, and each entry of AB can be queried in time $\mathcal{O}_d(\log \log n)$. Furthermore, for every $\varepsilon > 0$, the query time can be brought to constant time $\mathcal{O}(1/\varepsilon)$ if the running time is increased to near-linear $4^{d+o(d)}n^{1+\varepsilon}$. Notably, the running time is sublinear (essentially square root) in the number of (non-zero) entries.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms → Parameterized complexity and exact algorithms; Theory of computation → Logic → Finite Model Theory

Keywords and phrases Twin-width, matrices, parity and linear minors, model theory, linear algebra,

matrix multiplication, algorithms, computational complexity

Funding This paper is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 810115 – DYNASNET).

This paper was supported by the ANR projects (French National Research Agency) TWIN-WIDTH (ANR-21-CE48-0014-01) and Digraphs (ANR-19-CE48-0013-01).

1 Introduction

Since its introduction, the treewidth of a graph has proved to be a particularly important concept in graph theory, both in finite model theory [24], in algorithmic design (see for instance the textbook of Cygan et al. [16, Chapter 7]) and in structural analysis (see the Graph Minors series of Robertson and Seymour [38]). This invariant is strongly related to the concept of graph minor. Recall that a minor of a graph is a graph obtained by a succession of edge contractions and vertex or edge deletions. The treewidth of a graph is monotone with respect to this operation, in the sense that the treewidth of a minor of a graph G cannot be larger than the treewidth of G. By a classical theorem by Robertson and Seymour [39], a class of graphs has bounded treewidth if and only if its *minor closure* (that is, the set of all the minors of graphs in the class) does not contain all grids. In particular, a graph with huge treewidth admits a large square grid as a minor. This result, as well as its subsequent qualitative improvements (See [12], for instance), is the basis of the so-called *bidimensionality* algorithmic technique, a win-win argument leveraging low treewidth or the existence of a large grid minor [21].

This paper is the fifth of a series dedicated to a novel invariant of binary structures, the *twin-width* (See Section 2 for formal definitions). This invariant appeared to be particularly relevant for the study of *ordered* binary structures, and especially matrices over a finite alphabet [7]. Some of our results will only need the matrix entries to belong to a finite alphabet, while some will require the entries to belong to a finite field. A *submatrix* of a matrix is obtained by deleting some rows and columns. Most of our results concern (infinite) sets of matrices. In our framework, it will be natural to consider sets of matrices closed under the operation of taking a submatrix. Sets of matrices with this property are called *matrix classes*, analogously to permutation classes, which are classes of permutations closed under taking subpermutations. The alphabet or field being fixed (and having at least two elements), a matrix class is said to be *proper* if it does not include all matrices with entries in the prescribed alphabet or field.

The notion of a *rank Latin division* has been introduced in [7] (see definition in Section 5). It consists of a regular partition of the rows and columns delimiting blocks that either have constant entries or have full rank, in a globally controlled way, where full-rank blocks draw a universal permutation (see Figure 5). Just as the grids act as witnesses of a large treewidth, the rank Latin divisions witness a large twin-width: a matrix has either small twin-width or has a submatrix with a large rank Latin division. This is effective: in FPT time, either a contraction sequence of the matrix (witnessing that the twin-width is low) is output or a large rank Latin division is found in a submatrix (witnessing that the twin-width is high).

In this paper, we introduce an operation that plays a somewhat analogous role with respect to the twin-width of ordered binary structures that taking a minor plays with treewidth. Applied to 0,1-matrices, this operation consists of a succession of row or column deletions, and replacements of two consecutive rows or columns by their entry-wise sum (modulo 2).

Any 0,1-matrix obtained this way is a *parity minor* of the original matrix. More generally, when applied to matrices over a finite field $\mathbb{F}_p$, this operation consists of a succession of replacements of two consecutive rows or columns by a linear combination of these (over $\mathbb{F}_p$), and any matrix over $\mathbb{F}_p$ obtained this way is a *linear minor* of the original matrix. We say that a matrix class *excludes* a matrix M as a linear minor if M is not a linear minor of a matrix in the class. As expected, every matrix is a linear minor of any matrix having a sufficiently large rank Latin division, thus classes with unbounded twin-width do not exclude any matrix as a linear minor (Lemma 33). It appears that this necessary condition is also sufficient.

▶ **Theorem 1.** *A matrix class over a finite field has bounded twin-width if and only if it excludes some matrix as a linear minor.*

However, our proof of Theorem 1 involves some (finite) model theoretic arguments.

From a model theoretical point of view, a matrix over a finite alphabet of size p is seen as a structure with two linearly ordered sets of elements, the row and column index sets, and p binary predicates expressing the presence of a particular symbol at a specific entry. The logical formulas we will consider will allow distinguishing row and column indices, comparing indices of a same sort, and testing whether the entry of the matrix defined by two indices contains a given symbol.

It appears that the twin-width of ordered structured behaves very nicely with respect to first-order logic and (as we shall see) its modulo-counting extension. This situation is reminiscent of the relation of treewidth (and cliquewidth) with monadic second-order logic [14] and its modulo-counting extension [15].

Indeed, it follows from the results proved in the first paper of the series [8], that first-order model checking (that is: the problem of deciding whether a first-order (FO) sentence φ is satisfied on a structure) is fixed-parameter tractable on matrices over a finite alphabet, when parametrized by φ, the size of the alphabet, and the twin-width of the matrix, provided that some so-called *d-sequence* witnessing the upper bound on the twin-width is given together with the matrix. Here we observe that this result extends to the more expressive first-order logic with modulo-counting (FO+MOD), which is the logic obtained by adding to the standard first-order constructions new quantifiers $\exists^{i[p]}$, where "$\exists^{i[p]}x\ \varphi(x)$" expresses that the number of witnesses x for the formula φ is congruent to i modulo p.

Logical formulas also allow defining new structures from an original structure. This is the essence of the notion of transduction. A *transduction* of binary structures T first colors the elements of a given binary structure $\mathbf{A}$ in all possible ways, thus constructing a set of colored structures. Then, each of these colored structures gives rise to a new binary structure by means of fixed logical formulas, thus constructing a set $\mathsf{T}(\mathbf{A})$ of derived structures, the transduction of $\mathbf{A}$ by T. A set $\mathcal{D}$ of structures is a transduction of a set $\mathcal{C}$ of structures if there exists a transduction T with $\mathcal{D} \subseteq \mathsf{T}(\mathcal{C})$. A set $\mathcal{C}$ of structures is *monadically dependent* if the set of all finite graphs is not a transduction of $\mathcal{C}$. (While this actually follows from [3], we will take here this characteristic property as a definition of monadic dependence.) Note that it has been recently proved [11] that, for hereditary classes of structures (like matrix classes) monadic dependence coincides with the classical notion of dependence (or NIP), which is one of the most fundamental dividing lines in model theory; a proof of such a collapse in the particular case of hereditary classes of ordered graphs was previously shown in [7].

In [8], it was proved that for every FO-transduction T of binary structures, the maximum twin-width of a structure in $\mathsf{T}(\mathbf{A})$ is bounded by a function (depending on T) of the twin-width of $\mathbf{A}$. We also extend this result to FO+MOD-transductions (Theorem 27). As an example, there is a transduction L_p such that for every matrix M over $\mathbb{F}_p$, the set $\mathsf{L}_p(M)$ is

exactly the set of all linear minors of M (Lemma 29). Thus, the closure by linear minors of a matrix class with bounded twin-width also has bounded twin-width, from which Theorem 1 follows.

Together with the results established in [7], this leads to the following equivalence, where the equivalence with the properties in bold is proved in the current paper.

▶ **Theorem 2.** *Given a matrix class $\mathcal{M}$ over a finite field, the following are equivalent.*

(*i*) *$\mathcal{M}$ has bounded twin-width;*
(*ii*) ***$\mathcal{M}$ excludes a linear minor;***
(*iii*) *$\mathcal{M}$ is monadically dependent;*
(*iv*) *every matrix class that is an* FO*-transduction of $\mathcal{M}$ is proper;*
(*v*) ***every matrix class that is an* FO+MOD*-transduction of $\mathcal{M}$ is proper;***
(*vi*) *$\mathcal{M}$ is small (i.e. the number of $n \times n$ matrices in $\mathcal{M}$ is at most $2^{O(n)}$);*
(*vii*) ***every* FO+MOD*-transduction of $\mathcal{M}$ is small;***
(*viii*) *$\mathcal{M}$ is subfactorial (i.e. the number of $n \times n$ matrices in $\mathcal{M}$ is less than $n!$, for sufficiently large n).*

Assuming that FPT $\neq$ AW[$*$]*, those conditions are further equivalent to:*

(*ix*) FO*-model checking is* FPT *on $\mathcal{M}$;*
(*x*) FO+MOD***-model checking is* FPT *on*** $\mathcal{M}$.

We now consider some consequences of these results.

We call *rank-bidimensional* a parameterized problem defined on matrices whenever the presence of a large rank Latin division in a submatrix incurs an (easy) FPT algorithm. Thus, we get the following.

▶ **Theorem 3.** *Every* FO+MOD*-definable rank-bidimensional problem is in* FPT*.*

From Theorem 3, we obtain FPT algorithms for deciding if a (small) matrix is a linear minor (or parity minor) of another matrix, for exactly computing the grid number, mixed number, and grid rank of a matrix (see Section 2 for definitions).

Next we show that, over a finite field, the square M^2 of a matrix M with bounded twin-width has bounded twin-width, by expressing the squaring operation as an FO+MOD-transduction. From the characterization in terms of large rank Latin division of submatrices, it follows that if two matrices A and B have small twin-width, then so does the matrix $\left(\begin{smallmatrix} 0 & A \\ B & 0 \end{smallmatrix}\right)$. As $\left(\begin{smallmatrix} 0 & A \\ B & 0 \end{smallmatrix}\right)^2 = \left(\begin{smallmatrix} AB & 0 \\ 0 & BA \end{smallmatrix}\right)$, we deduce:

▶ **Theorem 4.** *There is a computable function $f : \mathbb{N}^2 \to \mathbb{N}$ such that the following holds. Let A and B be two conformal matrices over a finite field $\mathbb{F}_q$, both of twin-width at most d. Then the twin-width of the product AB is at most $f(d, q)$.*

Note that, by similar arguments, the sum of two conformal matrices over $\mathbb{F}_q$, both of twin-width at most d, has twin-width at most $f'(d, q)$, for some computable function f'.

We now consider the problem from an algorithmic point of view. From a computational perspective, the data structures used to encode matrices over a finite field are crucial. Encoding matrices as bipartite binary structures allows using the machinery developed for (ordered) graphs. In this setting, natural witnesses for twin-width boundedness are *d-sequences* (or, *contraction sequences*), which we mentioned earlier when discussing first-order model checking complexity on classes with bounded twin-width (see Section 2.2 for a formal definition). A naive implementation of the algorithm presented in [7, Theorem 2] runs in time $\exp(\exp(\mathcal{O}(d^2 \log d)))n^3$, and outputs a $2^{\mathcal{O}(d^4)}$-sequence if the twin-width is indeed at most d. We show how to bring the dependence in n down to $\mathcal{O}_d(n^2 \log n)$ (Theorem 11).

Gajarský et al. [23], building on Pilipczuk et al. [37], showed that given an n-vertex graph (or binary structure) G with a d-sequence and a first-order formula $\varphi(x_1, \dots, x_k)$, one can compute in time $\mathcal{O}_{d,\varphi}(n^{1+\varepsilon})$ (resp. $\mathcal{O}_{d,\varphi}(n)$) a data structure that answers for any query $v_1, \dots, v_k \in V(G)$ whether $G \models \varphi(v_1, \dots, v_k)$ in time $\mathcal{O}_{d,\varphi}(1/\varepsilon)$ (resp. $\mathcal{O}_{d,\varphi}(\log \log n)$). This result can be extended to FO+MOD. Then, the squaring operation can be performed by means of a simple interpretation, which gives a near-linear representation of M^2 (in the domain size, that is, sublinear in the number of matrix entries) where entries can be queried in constant time.

We thus obtain an algorithm that takes two matrices of bounded twin-width (*without* witnesses) and outputs their product in quasilinear time in the number of entries.

▶ **Theorem 5.** *Given two $n \times n$ matrices A and B over $\mathbb{F}_q$, both of twin-width at most d, there is an algorithm to compute their product AB in time $\mathcal{O}_{d,q}(n^2 \log n)$.*

However, this algorithm is not practical due to the acute dependence on the twin-width bound. We thus place ourselves in a setting where inputs are already in compact form (witnessing low twin-width). The use of an adapted internal representation is a classical technique of digital computing (Fourier transform, redundant representation of numbers, etc.). Likewise, it appears that convenient representations of matrices of bounded twin-width for matrix computations are *twin-decompositions* [5, 9]. Informally, a twin-decomposition is a tree whose leaves are bijectively mapped to the domain of the structure (here, to the row and column indices), and internal nodes are ordered and naturally correspond to contractions. The binary relations (here, the entries) are encoded by additional edges joining pairs of nodes of the tree, and respecting some specific rules. Every binary structure with bounded twin-width has a twin-decomposition with linearly many extra edges, hence the twin-decomposition forms a degenerate graph. The *width* of the twin-decomposition is related to this degeneracy (see Section 2 for precise definitions). Notice that a twin-decomposition of constant width takes quasilinear space to describe a set of binary relations with possibly quadratically many pairs. We show that a twin-decomposition can be computed from a contraction sequence in quadratic time and observe that, conversely, a contraction sequence can be computed from a twin-decomposition in linear time.

Our last contribution is an *ad hoc* efficient matrix multiplication algorithm for matrices over $\mathbb{F}_p$ with bounded twin-width, which we state here in the case of matrices over $\mathbb{F}_2$.

▶ **Theorem 6** (Theorem 7+[37])**.** *Let A and B be two $n \times n$ matrices over $\mathbb{F}_2$ given in the form of twin-decompositions of width at most d. For every $\varepsilon > 0$, there is a $4^{d+o(d)}n^{1+\varepsilon}$-time algorithm that outputs a twin-decomposition of the product AB of width $2^{d+o(d)}$ and a data structure of size $2^{d+o(d)}n^{1+\varepsilon}$ such that querying an entry of AB takes time $\mathcal{O}(1/\varepsilon)$.*

This result is based on an admittedly technical algorithm computing a twin-decomposition of the square M^2 of a matrix M from a twin-decomposition of M, which extends to a matrix multiplication algorithm for matrices each represented by a twin-decomposition.

▶ **Theorem 7.** *Let A and B be two $n \times n$ matrices over $\mathbb{F}_2$ given in the form of twin-decompositions of width at most d. There is a $4^{d+o(d)}n$-time algorithm that outputs a twin-decomposition of the product AB of width $2^{d+o(d)}$.*

The entries of AB can then be queried in time essentially the height of the twin-decomposition, which can be made logarithmic. However, by computing the data structure introduced by Pilipczuk et al. [37] in $\mathcal{O}(d'n^{1+\varepsilon})$ time and space where d' upperbounds the width of a twin-decomposition of the matrix, the entry queries can be performed in $\mathcal{O}(1/\varepsilon)$

time. Theorems 6 and 7 carry over on any finite field $\mathbb{F}_q$ with running time $q^{2d+o(d)}n$ and $2^{\mathcal{O}_q(d)}n^{1+\varepsilon}$, respectively.

An intriguing question concerns the existence of such results over infinite fields (starting with $\mathbb{Q}$). We do not have a direct definition of twin-width of matrices over $\mathbb{Q}$ based on contraction sequences. However linear-minor freeness naturally carries to infinite fields, and thus, it is natural to consider that a class of matrices over $\mathbb{Q}$ has bounded twin-width if its closure under linear minors is not the set of all matrices. This can be equivalently stated via the notion of *grid rank* of a matrix M, i.e., the largest k for which there is a $k \times k$ subdivision of M in which every block has rank at least k. Note that if a matrix has grid rank k, then any linear minor has grid rank at most k. Indeed, one can even show that a class of matrices has bounded grid rank if and only if it does not contain some matrix as a linear minor. We believe that computing the product of two matrices over $\mathbb{Q}$ with bounded grid rank should be done in almost quadratic time, however, we lack a structural decomposition as in the finite field case.

There is a vast literature on computing matrix multiplication, or other natural primitives of linear algebra, on classes of structured matrices. We give a few references on rank-structured matrices (see for instance [20, 27, 10, 41, 40]) and matrices of bounded treewidth.

A square matrix has quasiseparable order s if all its submatrices that are completely above the main diagonal, or completely below it, have rank at most s. Note that on adjacency matrices this is equivalent to the *linear rank-width* parameter (a dense analogue of pathwidth). Pernet [35] shows that multiplying two $n \times n$ matrices with quasiseparable order s can be done in time $\mathcal{O}(s^{\omega-2}n^2)$, where ω is the exponent of matrix multiplication, or $\mathcal{O}(s^3 n)$ if the matrices are given in a suitable compact form [36]. The closely-related semiseparable matrices also have efficient multiplication algorithms [42]. So-called[1] $\mathscr{H}$-matrices (for hierarchical matrices) and $\mathscr{H}^2$-matrices admit almost linear-time algorithms for vector-matrix multiplication [10].

One can naturally extend the treewidth graph parameter to 0,1-matrices M by considering the treewidth of the bipartite graph whose biadjacency matrix is M. Fomin et al. [22] show how to compute the determinant, the rank, and to solve a linear system defined by an $n \times n$ matrix of treewidth k, in time $k^{\mathcal{O}(1)}n$. It was recently shown by Dong et al. [18] how to solve linear programs in expected almost linear-time on matrices of bounded treewidth.

2 Preliminaries

We denote by $[i, j]$ the set of integers $\{i, i+1, \ldots, j-1, j\}$, and $[i]$ is a short-hand for $[1, i]$. We use the standard graph-theoretic notations: $V(G)$, $E(G)$, $N_G[S]$, $N_G(S)$, $G[S]$ respectively denote the vertex set, edge set, closed neighborhood of S, open neighborhood of S, and subgraph induced by S, of a graph G. Given a matrix M, we may interchangeably denote by $M_{x,y}$ or $M[x, y]$ the entry of M at row x and column y.

2.1 Binary structures and matrices

A relational *signature* Σ is a finite set of relation symbols R, each with a specified arity $r \in \mathbb{N}$. A *Σ-structure* $\mathbf{A}$ is defined by a set A (the *domain* of $\mathbf{A}$) together with a relation $R^{\mathbf{A}} \subseteq A^r$ for each relation symbol $R \in \Sigma$ with arity r. The syntax and semantics of first-order formulas over Σ, or *Σ-formulas* for brevity, are defined as usual. We will augment first-order logic

[1] These classes involve a hierarchical (dichotomic) block decomposition into blocks that are either of low rank (if far from the main diagonal) or recursively in the class (otherwise); see [27, 10] for definitions.

with modular counting quantifiers (see Section 4 for definitions). A *binary structure* is a Σ-structure such that every relation symbol of Σ has arity at most 2. An *ordered binary structure* is a structure $\mathbf{A}$ over a signature Σ consisting of unary and binary relation symbols which includes the symbol $<$, defining in $\mathbf{A}$ a total order on the domain of $\mathbf{A}$.

A matrix M over a finite alphabet A with rows R and columns C is viewed as an ordered binary structure with domain $R \uplus C$, equipped with the following relations:

- a unary relation R interpreted as the set of rows,
- an antisymmetric binary relation $<$ which defines a total order on $R \uplus C$, extending the total orders on the rows and columns of M in such a way that the rows precede the columns,
- one binary relation E_a, for each $a \in A$, where $E_a(r, c)$ holds if and only if r is a row, c is not (hence is a column), and a is the entry of M at row r and column c.

Matrices over a finite alphabet are actually as general as unconstrained binary structures. Indeed, one can encode any binary structure as an adjacency matrix M, where the values $M[x, y]$ are in one-to-one correspondence with the *atomic type of* (x, y), that is, the subset of unary relations containing x, the subset of unary relations containing y, and the subset of binary relations containing the (ordered) pair (u, v). The reader might think of matrices over finite alphabets, binary structures, or edge-colored graphs as different representations of the same objects. Depending on the context, we will sometimes prefer one representation over another.

2.2 Contraction sequences and twin-width

A *trigraph* G has vertex set $V(G)$, (black) edge set $E(G)$, and red edge set $R(G)$, with $E(G)$ and $R(G)$ being disjoint. The *set of neighbors* $N_G(v)$ of a vertex v in a trigraph G consists of all the vertices adjacent to v by a black or red edge. A d-trigraph is a trigraph G such that the *red graph* $(V(G), R(G))$ has degree at most d. In that case, we also say that the trigraph has *red degree* at most d. A (vertex) *contraction* or *identification* in a trigraph G consists of merging two (non-necessarily adjacent) vertices u and v into a single vertex z, and updating the edges of G in the following way. Every vertex of the symmetric difference $N_G(u) \triangle N_G(v)$ is linked to z by a red edge. Every vertex x of the intersection $N_G(u) \cap N_G(v)$ is linked to z by a black edge if both $ux \in E(G)$ and $vx \in E(G)$, and by a red edge otherwise. The rest of the edges (not incident to u or v) remain unchanged. We insist that the vertices u and v (together with the edges incident to these vertices) are removed from the trigraph. See Figure 1 for an illustration.

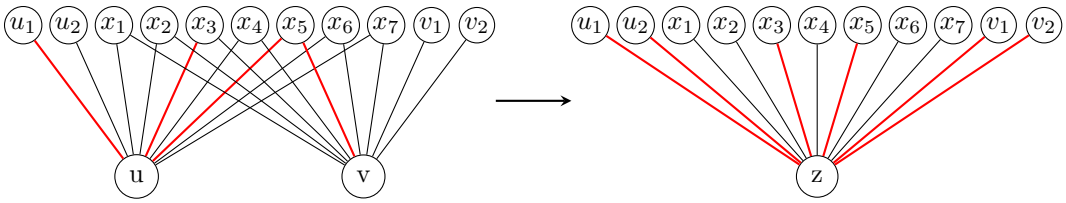

Figure 1 Contraction of vertices u and v, and how the edges of the trigraph are updated.

A *d-sequence* (or *contraction sequence*) is a sequence of d-trigraphs $G_n, G_{n-1}, \ldots, G_1$, where $G_n = G$, $G_1 = K_1$ is the graph on a single vertex, and G_{i-1} is obtained from G_i by performing a single contraction of two (non-necessarily adjacent) vertices. We observe that G_i has precisely i vertices, for every $i \in [n]$. The twin-width of G, denoted by $\mathrm{tww}(G)$, is the minimum integer d such that G admits a d-sequence. See Figure 2 for an illustration.

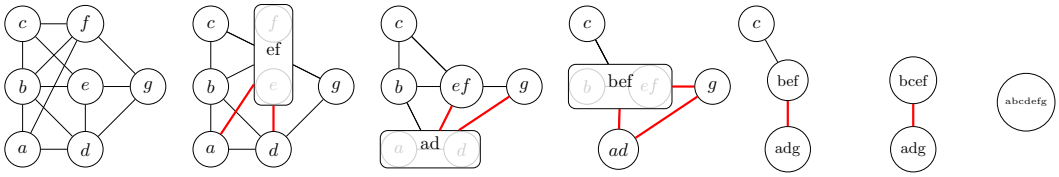

Figure 2 A 2-sequence witnessing that the initial graph has twin-width at most 2.

Twin-width can be generalized from graphs to binary structures in some (functionally) equivalent ways [8, 4]. Here we choose the following definition.

On general binary structures, red edges exist between two vertices $x, y \in V(G_i)$ whenever there are up to four vertices $u \neq v, u' \neq v' \in V(G)$ such that u and u' (which might be the same vertex) were contracted (together with possibly other vertices) to form x, similarly v and v' were contracted to form y, and the atomic types of (u, v) and of (u', v'), or of (v, u) and of (v', u'), are distinct. If instead, all such pairs (u, v) have the same atomic type, this shared atomic type labels the edge xy in G_i. Contraction sequences, d-sequences, and twin-width are then similarly defined.

In particular, we now have a definition of twin-width for the matrices of Section 2.1.

2.3 Matrix divisions

We will often denote by R (resp. C) the sets of row (resp. column) indices of a matrix M. For $X \subseteq R$ and $Y \subseteq C$, we denote by $M[X, Y]$ the submatrix of M consisting of the entries at rows in X and columns in Y.

A (k, ℓ)-*division* of a matrix M is a pair of partitions $(\mathcal{R} = \{R_1, \ldots, R_k\}, \mathcal{C} = \{C_1, \ldots, C_\ell\})$ of R and C, respectively, such that every R_i corresponds to consecutive rows, and every C_j, to consecutive columns. A k-*division* is a (k, k)-division. We may call a set of consecutive row or column indices an *interval*. The *grid number*, *mixed number*, *grid rank*, respectively, of a matrix is the largest integer k such that M has a k-division $(\{R_1, \ldots, R_k\}, \{C_1, \ldots, C_k\})$ for which, for every $i, j \in [k]$,

- $M[R_i, C_j]$ has a non-zero entry,
- $M[R_i, C_j]$ has at least two distinct rows and at least two distinct columns,
- $M[R_i, C_j]$ has at least k distinct rows or at least k distinct columns,

respectively.

The divisions are called *k-grid minor*, *k-mixed minor*, and *rank-k division*, respectively. An *interval minor* of a matrix M is a matrix N with k rows and ℓ columns such that M has a (k, ℓ)-division $(\{R_1, \ldots, R_k\}, \{C_1, \ldots, C_\ell\})$ such that for every $i \in [k]$ and every $j \in [\ell]$, $N_{i,j}$ is an entry of $M[R_i, C_j]$.

We call k-Grid Minor, k-Mixed Minor, Rank-k Division, Interval Minor Containment, respectively, the computational problems of deciding if an input matrix has grid number at least k, mixed number at least k, grid rank at least k, and if a matrix is an interval minor of another matrix.

The following celebrated theorem by Marcus and Tardos, solving the Füredi-Hajnal, and hence [29] the Stanley-Wilf conjectures, is crucial to the theory of twin-width.

▶ **Theorem 8** ([33]). *There is a function* $\mathrm{mt}\colon \mathbb{N} \to \mathbb{N}$ *such that for every natural* k*, every* $n \times m$ *matrix with at least* $\mathrm{mt}(k)\max(n, m)$ *non-zero entries admits a* k*-grid minor.*

The current best known bound is $\mathrm{mt}(k) = \frac{8}{3}(k+1)^2 2^{4k} = 2^{\mathcal{O}(k)}$ [13].

2.4 Computing contraction sequences

Efficiently (in polynomial time, FPT time, or even slice-wise polynomial time) approximating the twin-width of a binary structure remains a challenging open question. However, such an algorithm is known for totally ordered binary structures, or matrices over a finite alphabet.

▶ **Theorem 9** ([7]). *Given an $n \times m$ matrix M over a finite alphabet A, and an integer d, there is an $\exp(\exp(\mathcal{O}_{|A|}(d^2 \log d)))(n+m)^3$-time algorithm that*

- *either correctly reports that the twin-width of M is larger than d,*
- *or outputs a $|A|^{\mathcal{O}(d^4)}$-sequence for M.*

The exponent of $(n+m)$ in the algorithm running time was not made explicit in [7], but it can be observed that a direct implementation would indeed take cubic time. We will now see that we can bring this down to an almost quadratic dependence in the number of rows and columns. For the sake of convenience, we will work with symmetric matrices. For most applications, this is not really a limitation. To compute something on a non-symmetric matrix M, we will instead consider the symmetric $s(M) := \left(\begin{smallmatrix} 0 & M \\ M^T & 0 \end{smallmatrix}\right)$. Our main application concerns matrix multiplication. To compute the product MN of two square[2] matrices, we can compute

$$s(M)s(N^T) = \left(\begin{smallmatrix} 0 & M \\ M^T & 0 \end{smallmatrix}\right)\left(\begin{smallmatrix} 0 & N^T \\ N & 0 \end{smallmatrix}\right) = \left(\begin{smallmatrix} MN & 0 \\ 0 & M^T N^T \end{smallmatrix}\right).$$

That the resulting matrix is not necessarily symmetric is irrelevant. We read off MN from its top-left block.

We need the following definitions.

A k-division of a symmetric matrix is *symmetric* if for every $i \in [k]$, the i-th row part and the i-th column part contain the same row and column indices. Let M be a matrix and $\mathcal{D} = (\mathcal{R}, \mathcal{C})$ be a (k,ℓ)-division of M (with row set R, and column set C). A row part $R_i \in \mathcal{R}$ is *r-poor* in $\mathcal{D}$ if there is a union X of at most r column parts of $\mathcal{C}$ such that $M[R_i, C \setminus X]$ contains less than r distinct rows. An r-poor column part is defined symmetrically. In [7], a division $\mathcal{D}$ is said *r-rich* if none of its parts are r-poor.

A *symmetric fusion* of a matrix M with a symmetric k-division $(\mathcal{R}, \mathcal{C})$ consists of merging two consecutive row parts of $\mathcal{R}$, and the two corresponding column parts in $\mathcal{C}$. It results in a symmetric $(k-1)$-division of M. The notions of *(k,ℓ)-partition*, *k-partition*, and *symmetric merge* are obtained from *(k,ℓ)-division*, *k-division*, and *symmetric fusion*, respectively, by relaxing the condition that parts should consist of consecutive rows or columns.

As a preparatory step, we give an explicit lower bound for the grid rank to imply twin-width larger than d.

▶ **Lemma 10** ([7]). *Let d be a non-negative integer, and $k := 2d(d+1)+1$. If M admits a rank-k division, then M has twin-width larger than d.*

Proof. One can observe that a rank-k division is a $(k-1)$-rich division. Hence we assume that M has a $2d(d+1)$-rich division. The fact that M has then twin-width larger than d is the content of [6, Lemma 19]. ◀

Still following the previous paper of the series [7], we show the following.

▶ **Theorem 11.** *Given an $n \times n$ symmetric matrix M over a finite alphabet A, and an integer d, there is an algorithm running in time $\mathcal{O}_{d,|A|}(n^2 \log n)$ that*

[2] If not, one can pad them with rows or columns of 0 entries.

- *either correctly reports that the twin-width of M is larger than d,*
- *or outputs an* $\exp(\exp(\exp(\mathcal{O}_{|A|}(d^4))))$*-sequence for* M*.*

Proof. Let M be the input, as described, and set $k := 2d(d+1)+1$, and $r := |A|^{k \cdot \mathrm{mt}(k^2)}$. Let $\mathcal{D} = (\mathcal{R}, \mathcal{C})$ be the unique (symmetric) n-division of M, and $\mathcal{P} = (\mathcal{R}^{\mathcal{P}}, \mathcal{C}^{\mathcal{P}})$ be the unique (symmetric) n-partition of M. Thus, initially, $\mathcal{D} = \mathcal{P}$. Trivially, every part of $\mathcal{D}$ is r-poor in $\mathcal{D}$.

We need the following, shown in the previous paper of the series.

▷ Claim 12 ([6, Theorem 2, Lemma 20]). Checking if a row (resp. column) part R of size h is r-poor in $\mathcal{D}$ can be done in time $\exp(\exp_{|A|}(r \log r)) \cdot hn$.

If R is indeed r-poor, within the same time, a set Y of at most r column (resp. row) parts can be found, and we output a partition of R into the at most $r-1$ parts corresponding to the equivalent classes of the relation *being equal rows (resp. columns) outside* Y.

For convenience, if R is not r-poor, we output $\{R\}$.

Proof of the claim: This is direct from [6, Lemma 20]. The running time is indeed linear in the number of entries spanned by R, that is hn. ◊

Algorithm. While $\mathcal{D}$ has more than one row part, we do the following.

For every pair of consecutive row parts R_i, R_{i+1} in $\mathcal{R}$ with i odd, we check if $R_i \cup R_{i+1}$ is r-poor in $(\mathcal{R} \setminus \{R_i, R_{i+1}\} \cup \{R_i \cup R_{i+1}\}, \mathcal{C})$.
(A) If at least half of the pairs can be merged into an r-poor part, we make the corresponding symmetric fusions in $\mathcal{D}$. We obtain a new symmetric s-division $\mathcal{D}' = (\mathcal{R}', \mathcal{C}')$ where indices are rearranged from 1 to s, and a new symmetric partition $\mathcal{D}'^{\mathcal{P}} = (\mathcal{R}'^{\mathcal{P}}, \mathcal{C}'^{\mathcal{P}})$, which is the union of the partitions output by Claim 12.
(B) If, instead, less than half of the pairs can be merged, we will prove (slightly adapting [6, Section 5]) that the grid rank of M is large, which in turns implies that its twin-width is more than d, by Lemma 10. We thus exit the while loop.

Running time. Case (A) reduces by a factor of at least $1/4$ the current number of row parts in $\mathcal{D}$. Hence it can only be done $\mathcal{O}(\log n)$ times. By Claim 12, each such iteration takes time

$$\exp(\exp(r \log r)) \cdot \left(\sum_{i \text{ odd natural}} |R_i \cup R_{i+1}| \cdot n \right) = \mathcal{O}_d(n^2).$$

In this upper bound, i does not exceed the size of the current division, and the equality holds since the submatrices we consider partition M. The total running time of $\exp(\exp(\exp(\exp_{|A|}(d^4 \log d)))) \cdot n^2 \log n = \mathcal{O}_d(n^2 \log n)$ follows since case (B) stops the algorithm.

Correctness. It was shown in [7] that if (B) never occurs, the sequence of computed partitions $\mathcal{P}$ are coarser and coarser, and essentially form an $O_{d,|A|}(1)$-sequence: one can arbitrarily merge parts to bridge one partition to the next. Following the end of the proof of [6, Theorem 2, Case 2.], we obtain an $|A|^{\mathcal{O}(r^2)}$-sequence, hence an $\exp(\exp(\exp(\mathcal{O}_{|A|}(d^4))))$-sequence, for M. We thus focus on the case when (B) happens.

What was playing the role of case (B) in [7] (Case 1.) was more constrained: It stated that no two consecutive parts can be fused into an r-poor part. Then, one could directly show that, if this holds, the twin-width of M has to be large. Now, we only know that a positive fraction of the consecutive pairs cannot be fused into an r-poor part. We show that this still implies a large grid rank (following [6, Section 5]).

Let $\mathcal{D}^B = (\mathcal{R}^B = \{R_1^B, R_2^B, \dots, R_{2s-1}^B, R_{2s}^B\}, \mathcal{C}^B = \{C_1^B, C_2^B, \dots, C_{2s-1}^B, C_{2s}^B\})$ be the current division when (B) occurs. Note that, for convenience, we assumed that the number of row parts in $\mathcal{D}^B$ is even. We consider the symmetric division $\mathcal{D} = (\mathcal{R}, \mathcal{C})$ with

$$\mathcal{R} = \{R_1 = R_1^B \cup R_2^B, R_2 = R_3^B \cup R_4^B, \dots, R_s = R_{2s-1}^B, R_{2s}^B\},$$
$$\mathcal{C} = \{C_1 = C_1^B \cup C_2^B, C_2 = C_3^B \cup C_4^B, \dots, C_s = C_{2s-1}^B, C_{2s}^B\}.$$

At least half of the R_is (and symmetrically, of the C_is) are *not* r-poor.

Following [7, Theorem 22], in each column part of $\mathcal{C}$, we color in red the cells of $\mathcal{D}$ with at least k distinct rows or k distinct columns. We then color blue the cells X that feature a row vector which is not in a non-red cell, in the same column part of X and below it. In the at least $s/2$ column parts that are not r-poor, one can show that at least $\mathrm{mt}(k^2)$ cells are colored [7, Theorem 22]. We symmetrically fuse consecutive blocks of r-poor parts with an adjacent part which is not r-poor. Observe that the cells that are containing a formerly blue cell are now blue or red, and the cells containing a formerly red cell are still red. We now obtain a new symmetric division with at least $s/2$ column parts (or equivalently, row parts), where every column part contains at least $\mathrm{mt}(k^2)$ colored cells. Thus, we can finish the proof as in [7, Theorem 22], and obtain a rank-k division of M. Hence, by Lemma 10, the twin-width of M is larger than d. ◀

2.5 Twin-decompositions

A twin-decomposition of a graph G also uses the framework of a rooted carving decomposition, i.e., a rooted binary tree whose leaves are in one-to-one correspondence with the vertices of G. In the case of twin-decompositions though, the internal nodes of the rooted binary tree are totally ordered and the *width* is quite different from how carving-width is defined.

A rooted binary tree with $n-1$ internal nodes bijectively labeled by ℓ on $[n-1]$ is said *ranked* if whenever u and v are two distinct internal nodes such that v is a descendant of u, then $\ell(u) < \ell(v)$ holds. By convention, we then decide that all the leaves are labeled $+\infty$ (or equivalently n). For every $i \in [n]$, the *i-th border* of a ranked tree $\mathcal{T}$ with $n-1$ internal nodes is the set of maximal rooted subtrees whose roots have label at least i. We denote by $B_i(\mathcal{T})$ the i-th border of $\mathcal{T}$. It can be easily shown (using that $\mathcal{T}$ is a ranked tree) that the i-th border of $\mathcal{T}$ consists of exactly i subtrees. We denote by $r(\mathcal{T})$ the root of $\mathcal{T}$.

A *twin-decomposition* of an n-vertex graph G is a pair $(\mathcal{T}, \mathcal{B})$ where

(a) $\mathcal{T}$ is a rooted binary **t**ree, ranked by ℓ on $[n-1]$, whose leaves are in one-to-one correspondence with $V(G)$, and
(b) $\mathcal{B}$ (for **b**icliques) is a set of edges over $V(\mathcal{T})$ (disjoint from edge set of $\mathcal{T}$), such that:

1. $\mathcal{B}$ partitions the edge set of G, where an edge between $u, v \in V(\mathcal{T})$ is interpreted as the biclique of G linking every leaf in the subtree of $\mathcal{T}$ rooted at u, to every leaf in the subtree of $\mathcal{T}$ rooted at v, and
2. No edge of $\mathcal{B}$ *crosses* an i-th border, i.e., links a node in a subtree $\mathcal{T}'$ of $B_i(\mathcal{T})$ but *not* the root of $\mathcal{T}'$ to a vertex outside every subtree of $B_i(\mathcal{T})$.

One can retrieve the sequence of trigraphs $G = G_n, \dots, G_1$ from the twin-decomposition in the following way. The vertex set of G_i corresponds to the subtrees of $B_i(\mathcal{T})$,

- with a black edge between $\mathcal{T}_1 \in B_i(\mathcal{T})$ and $\mathcal{T}_2 \in B_i(\mathcal{T})$ whenever there is an edge in $\mathcal{B}$ between $r(\mathcal{T}_1)$ or one of its ancestors (in $\mathcal{T}$) and $r(\mathcal{T}_2)$ or one of its ancestors, or whenever for every leaf u of $\mathcal{T}_1$ and every leaf v of $\mathcal{T}_2$, there is an edge between the path from u to $r(\mathcal{T}_1)$ and the path from v to $r(\mathcal{T}_2)$,

- and a red edge between $\mathcal{T}_1 \in B_i(\mathcal{T})$ and $\mathcal{T}_2 \in B_i(\mathcal{T})$ whenever this does not hold but yet there is an edge $uv \in \mathcal{B}$ with $u \in V(\mathcal{T}_1)$ and $v \in V(\mathcal{T}_2)$, and a pair (u', v') such that u' is a leaf of $\mathcal{T}_1$, v' is a leaf of $\mathcal{T}_2$, and there is no edge between the path from u' to $r(\mathcal{T}_1)$ and the path from v' to $r(\mathcal{T}_2)$.

The width of the twin-decomposition $(\mathcal{T}, \mathcal{B})$ is again defined as the maximum red degree among every vertex of every trigraph G_i (as previously defined). See Figure 3 for an illustration of a twin-decomposition corresponding to a particular contraction sequence.

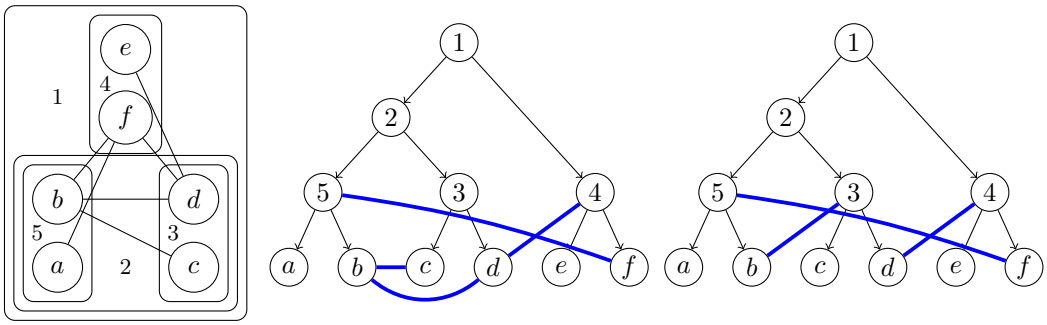

Figure 3 Left: a graph G with a contraction sequence (or partition sequence), where trigraph G_i is obtained after performing the contraction labeled i. Center: the twin-decomposition corresponding to this contraction sequence, with the edges of $\mathcal{B}$ in blue. Right: a ranked tree $\mathcal{T}$ and a partition $\mathcal{B}$ of the edges of G that does *not* make for a twin-decomposition, since the edge $b3$ crosses $B_5(\mathcal{T})$ (and $B_4(\mathcal{T})$).

Suppose $(\mathcal{T}, \mathcal{B})$ is a twin-decomposition of a graph with width at most d. If one orients every edge $uv \in \mathcal{B}$ as the arc (u, v) whenever the parent of u has a larger label than the parent of v (keeping the edge undirected if u and v shares the same parent), then every node z of $V(\mathcal{T})$ has at most d out-neighbors. Indeed at the i-th border, with i being the label of the parent $p(z)$ of z, the subtree of $B_i(\mathcal{T})$ rooted at $p(z)$ has red degree at most one per out-neighbor of z. This is because two neighbors of z (in $\mathcal{B}$) cannot be in a descendant-ancestor relationship, as we imposed $\mathcal{B}$ to *partition* the edge set of G. Since z has at most one undirected edge incident to it, the graph made by $\mathcal{B}$ on $V(\mathcal{T})$ is $d+1$-degenerate.

Thus the mere definition of twin-decomposition imposes that $|\mathcal{B}| = \mathcal{O}(n)$.

For every twin-decomposition $(\mathcal{T}, \mathcal{B})$, we say that $\mathcal{B}$ is *lifted up* if for every $uv \in \mathcal{B}$ such that $\ell(p(u)) \geqslant \ell(p(v))$, either $p(v) = p(u)$ or if $u' \in V(\mathcal{T}) \setminus \{u\}$ is such that $p(u') = p(u)$, then $u'v \notin \mathcal{B}$. For example, the edge set of the twin-decomposition in the center of Figure 3 is lifted up.

▶ Remark 13. For every twin-decomposition $(\mathcal{T}, \mathcal{B})$ of width d, one can compute in time $\mathcal{O}(dn)$ a set $\mathcal{B}' \subseteq E(\mathcal{T})$ of size at most $|\mathcal{B}|$ such that $(\mathcal{T}, \mathcal{B}')$ is a twin-decomposition of the same sequence where $\mathcal{B}'$ is lifted up. Indeed, for each i ranging from $n-1$ to 1, if $z = \ell^{-1}(i)$ and $u, u' \in V(\mathcal{T})$ are such that $p(u) = p(u') = z$, for each $v \in V(\mathcal{T})$ such that $uv, u'v \in \mathcal{B}$, we replace $\mathcal{B}$ by $(\mathcal{B} \setminus \{uv, u'v\}) \cup \{zv\}$.

▶ Remark 14. If $(\mathcal{T}, \mathcal{B})$ is a twin-decomposition where $\mathcal{B}$ is lifted up, then the edges of $\mathcal{B}$ are exactly the black edges "disappearing" at some point of the contraction sequence associated to $\mathcal{T}$. More precisely, $e = uv \in \mathcal{B}$ if and only if there exists an i such that u and v are distinct vertices of $V(G_i)$, $uv \in E(G_i)$ and either u and v are identified in G_{i-1} or $u'v' \in R(G_{i-1})$, where u' and v' denote the vertices of G_{i-1} that contain respectively u and v.

As previously observed, twin-decompositions permit to retrieve their corresponding contraction sequences, i.e., adjacencies and red adjacencies in the G_is. However, given a

twin-decomposition $(\mathcal{T}, \mathcal{B})$ of G such that $|\mathcal{B}| = \mathcal{O}(n)$, we should not store all the $R(G_i)$ for each $i \in [n-1]$ as it may take $\Omega(n^2)$ space; too much for our algorithmic purposes when we want a linear complexity in n (see Section 7). To do so, we will need to compute $R(G_i)$ on the fly.

Assume that $G_n, \dots, G_1$ is a d-sequence of G, with corresponding twin-decomposition $(\mathcal{T}, \mathcal{B})$. Given $(\mathcal{T}, \mathcal{B})$, we show how to compute all the red neighborhoods $\{W \in V(G_i) : UW \in R(G_i)\}$ for every $i \in [n-1], U \in V(G_i)$ in $\mathcal{O}(dn)$ time. Note that we may identify the vertices of $V(G_i)$ with their corresponding subsets of vertices of $V(G)$ (and thus denote them with uppercase symbols). For each $i \in [n-1], U \in V(G_i)$, we let $L^i_U := \{W_1, \dots, W_k\}$ be the $k \leqslant d$ red neighbors of U in G_i. We say that the L^i_U can be *dynamically computed* if there exists an algorithm which runs in n steps, such that at the end of step i (where i ranges from $n-1$ to 1), it only stores in memory the values of L^i_U for each $U \in V(G_i)$, and every step takes time $\mathcal{O}_d(1)$.

▶ **Lemma 15.** *Given a twin-decomposition $(\mathcal{T}, \mathcal{B})$ of width d of an n-vertex graph G, one can dynamically compute in time $\mathcal{O}(dn)$ the adjacency lists L^i_U of each node of $V(\mathcal{T})$.*

Proof. By Remark 13 we may assume that $\mathcal{B}$ is lifted up. We simply need to process the nodes of $\mathcal{T}$ by decreasing values of their labels and update in constant time the lists L^i_U according to the information given by $\mathcal{B}$. The formal algorithm is given by Algorithm 1. Since the width of the twin-decomposition is at most d, at each step, the number of considered edges of $\mathcal{B}$ is at most d, which gives the desired complexity. In the following algorithm, the set $\mathcal{V}$ at step i correspond to $V(G_i)$ and the list L^i_U is given by the content of L_U at the end of step i. ◀

Algorithm 1 On-the-fly computation of the red graphs based on a twin-decomposition

Input : A twin-decomposition $(\mathcal{T}, \mathcal{B})$ of width d.
$\mathcal{V} \leftarrow \bigcup_{u \in V(G)} \{\{u\}\}$
for $U \in \mathcal{V}$ **do**
  $L_U \leftarrow \{\}$
for $i = n-1 \to 1$ **do**
  Let Z be the node of $\mathcal{T}$ labeled by i and U, V denote its two children in $\mathcal{T}$.
  $\mathcal{V} \leftarrow \mathcal{V} \setminus \{U, V\} \cup \{Z\}$
  $L_Z \leftarrow L_U \cup L_V \setminus \{U, V\}$
  for $W \in \mathcal{V} \setminus \{Z\}$ *such that* $VW \in \mathcal{B}$ *xor* $UW \in \mathcal{B}$ **do**
    $L_Z \leftarrow L_Z \cup \{W\}$

For the sake of simplicity, we described twin-decompositions for graphs but they readily generalize to binary structures. Here, the representation of binary structures with edge-colored graphs will be the most convenient.

We thus see a binary structure as a graph G with an edge-labeling function $\nu : E(G) \to \mathbb{F}_q$. A twin-decomposition of G is still a pair $(\mathcal{T}, \mathcal{B})$, only now, edges of $\mathcal{B}$ are labeled over $\mathbb{F}_q$. For every $\ell \in \mathbb{F}_q$, let G_ℓ denote the subgraph of G induced by the set of edges labeled ℓ, and by $\mathcal{B}_\ell$, the subset of $\mathcal{B}$ of edges labeled ℓ. Then $(\mathcal{T}, \mathcal{B}_\ell)$ is a twin-decomposition of G_ℓ. We say that $\mathcal{B}$ is lifted up if $\mathcal{B}_\ell$ is lifted up for every $\ell \in \mathbb{F}_q$.

2.6 Computing a twin-decomposition from a contraction sequence

There is an easy linear (in the input size, i.e., in the number of edges) algorithm that, given a d-sequence, computes a corresponding twin-decomposition (of same width d). The *list of triples* of a contraction sequence $G_n, \dots, G_i, \dots, G_1$ is $(u_n, v_n, z_n), \dots, (u_i, v_i, z_i), \dots, (u_2, v_2, z_2)$ such that the contraction of u_i and v_i in G_i into a new vertex z_i results in G_{i-1}.

▶ **Theorem 16.** *A twin-decomposition of width d of an n-vertex graph G given with the list of triples of a d-sequence can be computed in time $\mathcal{O}(n^2)$.*

Proof. Let $(u_n, v_n, z_n), \dots, (u_2, v_2, z_2)$ be the list of triples of the given d-sequence of G. We want to build a twin-decomposition $(\mathcal{T}, \mathcal{B})$ of width d for G. We initialize $\mathcal{T}$ by creating one leaf for each vertex of G, each leaf being labeled by the corresponding vertex. We initialize $\mathcal{B}$ as the empty set.

We process the list of triples from (u_n, v_n, z_n) to (u_2, v_2, z_2). At the i-th iteration (for some $i \in [2, n]$) we explicitly perform the contraction of u_i and v_i in G_i (into z_i), and obtain the new trigraph G_{i-1}. In $\mathcal{T}$, we add a common parent z_i to u_i and v_i, and set $\ell(z_i) = i - 1$. For every $w \in V(G_{i-1}) \setminus \{z_i\}$ such that $\{u_i w, v_i w\} \cap E(G_i) \neq \emptyset$ and $z_i w \in R(G_{i-1})$, we add the edge $u_i w$ to $\mathcal{B}$ if $u_i w \in E(G_i)$, or we add the edge $v_i w$ to $\mathcal{B}$ if $v_i w \in E(G_i)$. One can observe that the i-th step takes $\mathcal{O}(n)$ time, hence the overall running time is quadratic. ◀

Actually, on an n-vertex m-edge graph the twin-decomposition of a d-sequence of G can be computed in time $\mathcal{O}(dn + m)$.

▶ **Observation 17.** *There is an $\mathcal{O}(n)$-time algorithm that inputs a twin-decomposition $(\mathcal{T}, \mathcal{B})$ of width d of an n-vertex graph G, and outputs the list of triples of a d-sequence of G.*

Proof. We add identifying labels, say *tags*, to the nodes of $\mathcal{T}$ (in addition to the labeling function ℓ). The tags of the leaves match the one-to-one correspondence between $V(G)$ and the leaves of $V(T)$. For every i from n down to 2, append the triple (u_i, v_i, z_i) where u_i and v_i are the tags of the two children of the node $v = \ell^{-1}(i-1)$, and tag v with the fresh identifier z_i. ◀

A consequence of the second paper of the series is that, given twin-decompositions of bounded width, one can find twin-decompositions of (larger) bounded width, where in addition the tree $\mathcal{T}$ has logarithmic depth.

▶ **Theorem 18.** [4, see Lemma 23 and Proposition 22] *For every integer d, there is a larger integer D such that every n-vertex graph of twin-width d admits a twin-decomposition of width at most D and depth $\mathcal{O}_d(\log n)$.*

Given a twin-decomposition $(\mathcal{T}, \mathcal{B})$ of G with width d and depth h, the presence of an edge between two vertices of G can be decided in time $\mathcal{O}(dh)$. This yields a linear-space representation of G with edge queries in logarithmic time. The following stronger result was shown by Pilipczuk et al.

▶ **Theorem 19** ([37])**.** *Given a twin-decomposition of width d of an n-vertex graph G, and any $\varepsilon > 0$, there is a data structure of size $\mathcal{O}(dn^{1+\varepsilon})$ (resp. $\mathcal{O}_d(n)$), computable in time $\mathcal{O}(dn^{1+\varepsilon})$ (resp. $\mathcal{O}_d(n \log n \log\log n)$), that supports edge queries of G in time $\mathcal{O}(1/\varepsilon)$ (resp. $\mathcal{O}_d(\log\log n)$).*

All the results mentioned in this section extend from graphs to binary structures.

2.7 Parameterized complexity of model checking

First-order (FO) matrix model checking asks, given a matrix M (or a totally ordered binary structure $\mathcal{S}$) and a first-order sentence φ (i.e., a formula without any free variable), if $M \models \varphi$ holds, that is, if φ is true in M. FO model checking is fixed-parameter tractable (FPT) on a matrix class $\mathcal{M}$, with respect to the sentence size and the input matrix, if there exists a constant c and a computable function f, such that $M \models \varphi$ can be decided in time $f(|\varphi|)\,(m+n)^c$, for every $n \times m$ matrix $M \in \mathcal{M}$ and FO sentence φ.

FO model checking of general (unordered) graphs is AW[$*$]-complete [19], and thus very unlikely to be FPT. Indeed FPT $\neq$ AW[$*$] is a much weaker assumption than the already widely-believed Exponential Time Hypothesis [28], and if false, would in particular imply the existence of a subexponential algorithm solving 3-SAT. FO model checking of general binary structures of bounded twin-width given with an $\mathcal{O}(1)$-sequence can even be solved in linear FPT time $f(|\varphi|)\,|U|$, where U is the domain of the structure [8].

Gajarský et al. [23] reproved that result using a different, and more standard formalism. Building on Theorem 19, they also presented an algorithm that inputs a binary structure given with an $\mathcal{O}(1)$-sequence and a formula with some free variables, and after some linear-time processing, can answer queries of the form *does the given tuple satisfy the formula in the structure* in doubly-logarithmic time.

▶ **Theorem 20** ([23]). *Given a binary Σ-structure $\mathbf{A}$ on a domain of size n, a d-sequence of $\mathbf{A}$, and a first-order Σ-formula $\varphi(x_1,\dots,x_k)$, there is a data structure computable in time $\mathcal{O}_{d,\varphi}(n)$ that given any query of the form $v_1,\dots,v_k \in A$ reports in time $\mathcal{O}_{d,\varphi}(\log\log n)$ whether $\mathbf{A} \models \varphi(v_1,\dots,v_k)$ holds.*

Up to increasing the preprocessing time to near-linear, the queries can even be met in constant time.

▶ **Theorem 21** ([23]). *For every $\varepsilon > 0$, given a binary Σ-structure $\mathbf{A}$ on a domain of size n, a d-sequence of $\mathbf{A}$, and a first-order Σ-formula $\varphi(x_1,\dots,x_k)$, there is a data structure computable in time $\mathcal{O}_{d,\varphi}(n^{1+\varepsilon})$ that given any query of the form $v_1,\dots,v_k \in A$ reports in time $\mathcal{O}_{d,\varphi}(1/\varepsilon)$ whether $\mathbf{A} \models \varphi(v_1,\dots,v_k)$ holds.*

In classes of ordered binary structures or matrices, one need not require that the contraction sequence is given in input. Indeed there is an FPT approximation algorithm, that takes a matrix M of twin-width d, and outputs a $g(d)$-sequence of M in time $h(d)\,|M|^{\mathcal{O}(1)}$ [7]. Hence, FO matrix model checking can be solved in FPT time $f(|\varphi|)\,|M|^{\mathcal{O}(1)}$ [7] in classes of bounded twin-width.

2.8 Interpretations and transductions

Let Σ, Γ be relational signatures. A *(simple) interpretation* $\mathsf{I}\colon \Sigma \to \Gamma$ consists of the following Σ-formulas: a *domain* formula $\nu(x)$, and for each relation symbol $R \in \Gamma$ of arity r, a formula $\rho_R(x_1,\dots,x_r)$. If $\mathbf{A}$ is a Σ-structure, the Γ-structure $\mathsf{I}(\mathbf{A})$ has domain $\nu(\mathbf{A}) = \{v \in A : \mathbf{A} \models \nu(v)\}$ and the interpretation of a relation symbol $R \in \Sigma$ of arity r is $\rho_R(\mathbf{A}) \cap \nu(\mathbf{A})^r$, that is:

$$R^{\mathsf{I}(\mathbf{A})} = \{(v_1,\dots,v_r) \in \nu(\mathbf{A})^r : \mathbf{A} \models \rho_R(v_1,\dots,v_r)\}.$$

Note that Theorems 20 and 21 can be seen as efficiently computing the structure $\mathsf{I}(\mathbf{A})$ given the formulas $\rho_R(x_1,\dots,x_r)$, when $\mathbf{A}$ has bounded twin-width and is given with a contraction

sequence. If $\mathcal{C}$ is a set of Σ-structures then denote $\mathsf{I}(\mathcal{C}) = \{\mathsf{I}(\mathbf{A}) \mid \mathbf{A} \in \mathcal{C}\}$. A class $\mathcal{C}$ *interprets* a class $\mathcal{D}$ if there is an interpretation I such that $\mathsf{I}(\mathcal{C}) \supseteq \mathcal{D}$.

Let $\Sigma \subseteq \Sigma^+$ be relational signatures. The *Σ-reduct* of a Σ^+-structure $\mathbf{A}$ is the structure obtained from $\mathbf{A}$ by "forgetting" all the relations not in Σ; we denote this interpretation as $\mathsf{Reduct}_\Sigma \colon \Sigma^+ \to \Sigma$. A class $\mathcal{C}$ of Σ-structures *transduces* a class $\mathcal{D}$ if there is a class $\mathcal{C}^+$ of Σ^+-structures, where Σ^+ is the union of Σ and some unary relation symbols such that $\mathsf{Reduct}_\Sigma(\mathcal{C}^+) = \mathcal{C}$ and $\mathcal{C}^+$ interprets $\mathcal{D}$. Here, the unary relation symbols and the underlying interpretation are called *transduction* (or FO-transduction).

As it will be enough for our purposes, we will here take the following characterization as the definition of monadic dependence.

▶ **Theorem 22** (Baldwin and Shelah [3]). *A class $\mathcal{C}$ of Σ-structures is monadically dependent if and only if $\mathcal{C}$ does not transduce the set of all finite graphs.*

Importantly transductions preserve the boundedness of twin-width.

▶ **Theorem 23** ([8]). *Let $\mathcal{C}$ be a set of Σ-structures of bounded twin-width, and $\mathsf{T} \colon \Sigma \to \Sigma$ be a transduction. Then $\mathsf{T}(\mathcal{C})$ has bounded twin-width.*

We will deal with FO+MOD-transductions, which are defined as FO-transductions but with first-order logic augmented with modular counting quantifiers. We postpone the relevant definitions to Section 4.

2.9 Organization

In Section 3, we introduce the parity and linear minors. In Section 4, we define the logic FO+MOD and show that parity and linear minors can be expressed with FO+MOD-transductions. In Section 5, we use the two previous sections to show Theorem 1, that is, the equivalence between bounded twin-width and linear-minor freeness. In Section 6, we introduce the rank bidimensionality, and classify several matrix problems involving a division of the rows and columns as being fixed-parameter tractable. In Section 7, we use the extensions of Section 4 to show that bounded twin-width is preserved by matrix product, and give an almost quadratic algorithm to multiply two matrices of bounded twin-width (Theorem 5). In Section 8, we present a linear-time (i.e., possibly sublinear in the number of non-zero entries) algorithm when further given a twin-decomposition of the two matrices to multiply.

3 Parity and linear minors

We now introduce the notion of *parity minor* for matrices over a finite field. Let M be a matrix with entries in a field $\mathbb{F}$. A *deletion operation* (or simply *deletion*) in M consists of deleting a row or a column. A *sum operation* (or simply *sum*) in M consists of replacing any pair of *consecutive* rows r_i, r_{i+1} or columns c_j, c_{j+1} by their pointwise sum in $\mathbb{F}$. That is, r_{i+1} is deleted and r_i is replaced by r'_i with $r'_i[k] = r_i[k] + r_{i+1}[k]$, for every column index k, where $+$ is the addition in $\mathbb{F}$. We say that a matrix N is a *parity minor* of M, denoted by $N \leqslant_{\mathrm{pm}} M$, if N can be obtained from M by a sequence of deletions and sums.

▶ **Observation 24.** *If N is a parity minor of M, then N can be obtained from M by performing all the deletions before performing all the sums.*

Proof. Any (single) deletion performed just after a (single) sum can be equivalently performed just before the sum. If the deletion is precisely on the row or column of the sum, one can equivalently remove the two consecutive rows or columns and not perform the sum. ◀

On the contrary, as we impose the sums to be on consecutive rows or columns, one *cannot* necessarily perform all the sums before all the deletions.

Observation 24 implies the following reformulation. A *parity minor* of a matrix M is any $n \times m$ matrix obtained by summing up every cell of an (n,m)-division of a submatrix of M. Indeed, after deleting the rows and columns not part of the submatrix, one can obtain the minor by summing every row and column part of the (n,m)-division into a single row and column. This equivalent definition justifies the term of "parity minor"; Over the binary field $\mathbb{F}_2$, the minor operation boils down to dividing a submatrix and keeping from each cell its parity of 1-entries. See Figure 4 for an illustration.

$$M = \left[\begin{array}{cccccccc} 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right] \qquad \left[\begin{array}{cc|cccc|cc} 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ \hline 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right] \qquad N = \left[\begin{array}{ccc} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{array}\right]$$

Figure 4 A parity minor, equivalently linear minor, N of a matrix M over $\mathbb{F}_2$. In the middle, the deleted rows and columns of M are in light gray, while the $(4,3)$-division of the remaining submatrix giving rise to N is represented with solid black lines.

We now define a more permissive notion of *linear minor* by replacing the sum operation by a weighted sum, or linear combination. A *weighted sum operation* (or simply *weighted sum*) in M consists of replacing any pair of consecutive rows r_i, r_{i+1} (or columns c_j, c_{j+1}) by $\alpha r_i + \beta r_{i+1}$ (or $\alpha c_j + \beta c_{j+1}$) for some chosen $\alpha, \beta \in \mathbb{F}$. Again it means that r_{i+1} is deleted and r_i is replaced by r'_i with $r'_i[k] = \alpha r_i[k] + \beta r_{i+1}[k]$, for every column index k, where operations are performed in $\mathbb{F}$. Choosing $\alpha = 1_\mathbb{F}$ and $\beta = 0_\mathbb{F}$ emulates the deletion of r_{i+1}, while $\alpha = 0_\mathbb{F}$ and $\beta = 1_\mathbb{F}$ corresponds to the deletion of r_i. Thus we no longer need to add the deletion operations. We say that a matrix N is a linear minor of M, denoted by $N \leqslant_{\mathrm{lm}} M$, if N can be obtained from M by a sequence of weighted sums. Let us call $\mathbb{F}$-*weighting of* M (or simply *weighting of* M if $\mathbb{F}$ is clear from the context) any mapping $w : \mathrm{rows}(M) \cup \mathrm{cols}(M) \to \mathbb{F}$. Equivalently a linear minor of a matrix M over a finite field $\mathbb{F}$ is any $n \times m$ matrix N obtained from an $\mathbb{F}$-weighting w and (n,m)-division $\mathcal{D} = (\{R_1, \ldots, R_n\}, \{C_1, \ldots, C_m\})$ of M by replacing every cell $M[R_i, C_j]$ of $\mathcal{D}$ by the single entry $\sum_{r \in R_i, c \in C_j} w(r)w(c)M_{r,c}$, that is, setting $N_{i,j} = \sum_{r \in R_i, c \in C_j} w(r)w(c)M_{r,c}$. We can indeed rewrite the entries of N that way, since every finite field is commutative [32].

The second definition actually works for any commutative field, while the first one accommodates any field. Yet another way to see a linear minor over a commutative field is that one first divides (i.e., partitions into intervals) the column set of M, picks one column vector in the span of each column part of M to form the matrix M', divides now the row set of M', and finally picks one row vector in the span of each row part of M' to obtain N a linear minor of M. Of course the row division may precede the column division instead. We observe that over the binary field the only possible weighted sums are deletions and (simple) sums.

▶ **Observation 25.** *Over $\mathbb{F}_2$, parity minors and linear minors coincide.*

We recall that a matrix class is a set of matrices which is closed under taking submatrices. The *parity-minor closure* of $\mathcal{M}$, denoted by $\mathrm{Clos}_{\mathrm{pm}}(\mathcal{M})$, is the matrix class of all matrices

N which are parity minors of some M in $\mathcal{M}$. Similarly, the *linear-minor closure* of $\mathcal{M}$, denoted by $\mathrm{Clos}_{\mathrm{lm}}(\mathcal{M})$, is the matrix class of all matrices N which are linear minors of some M in $\mathcal{M}$. We say that $\mathcal{M}$ is *parity-minor free* (resp. *linear-minor free*) if its parity-minor closure (resp. linear-minor closure) is not the set of all $\mathbb{F}$-matrices. We may denote the set of all $\mathbb{F}$-matrices by $\mathcal{M}(\mathbb{F})$, or by $\mathcal{M}_{\mathrm{all}}$ if the field $\mathbb{F}$ is clear from the context. Thus $\mathcal{M}$ is linear-minor free over $\mathbb{F}$ if $\mathrm{Clos}_{\mathrm{lm}}(\mathcal{M}) := \{N \ : \ N \leqslant_{\mathrm{lm}} M,\ M \in \mathcal{M}\} \neq \mathcal{M}(\mathbb{F})$. We also denote by $\mathcal{M}_n(\mathbb{F})$ the set of $n \times n$ matrices over $\mathbb{F}$, and by $\mathcal{M}_\square(\mathbb{F})$ the set of square matrices over $\mathbb{F}$.

4 First-order logic with modular counting

First-order logic with modular counting, denoted by FO+MOD, augments FO with a new kind of quantifiers, $\exists^{i[p]}$ for some fixed integer p, and any $i \in [0, p-1]$. The semantics is extended as follows. For any structure $\mathcal{M}$ with domain M, $\mathcal{M} \models \exists^{i[p]} x\ \varphi(x)$ holds whenever $|\{a \in M \mid \mathcal{M} \models \varphi(a)\}| \equiv i \mod p$. Informally, one can now express that the number of witnesses for a formula $\varphi(x)$ is equal to i modulo p. We may use $\exists^e$ for $\exists^{0[2]}$, and $\exists^o$ for $\exists^{1[2]}$. As for FO-sentences, an FO+MOD-sentence is said *prenex* if it is formed by a succession of quantifiers followed by a quantifier-free formula. Every sentence of quantifier rank ℓ is logically equivalent to a prenex sentence of quantifier rank $h(\ell)$ for some tower function h.

The following theorem (Theorem 26) was proven in [8] with FO-sentences instead of FO+MOD-sentences, and follows from that paper with a small adaptation. It can also be observed from the work of Gajarský et al. [23]. Here we choose to adapt the proof from the first paper of the series [8]. In both cases, making the proof self-contained would require quite a lot of background and overlap with either one of those papers, and in the end, would not bring anything new. A reader interested in the proof will first have to read Section 7 of [8], and in particular, get familiar with morphism-trees (similar to Ehrenfeucht-Fraïssé game trees) and the dynamic-programming algorithm handling them.

▶ **Theorem 26** (follows with some small adjustments from [8]). *Given a d-sequence of a binary structure $\mathcal{S}$, and a prenex* FO+MOD*-sentence φ of quantifier rank ℓ, one can decide $\mathcal{S} \models \varphi$ in time $f(\ell, d)n$ for some computable non-elementary function f.*

Proof. As mentioned above, this essentially follows from the FO model-checking algorithm presented in [8, Section 7]. However we need to slightly modify the way we reduce the morphism-trees to account for the modular counting quantifiers. We do not reduce two equivalent nodes, but rather, if we find $p+1$ pairwise equivalent nodes, we only keep one such node. One can observe that the size of a reduct of any depth-ℓ morphism-tree is still bounded by a function of ℓ (since p is an absolute constant).

This is the only modification to the dynamic-programming updates of the theories local to the red graph. At every node of $MT'_\ell(G, \mathcal{P}_i, X)$, the existence, universality, and number of witnesses modulo p among the parts with local root X is preserved (regardless of the prenex sentence of quantifier depth ℓ). ◀

A related result by Kuske and Schweikardt [30] asserts that FO+MOD model checking is FPT in classes of locally bounded treewidth.

Another useful fact is that FO-transductions of bounded twin-width classes have themselves bounded twin-width (see [8, Section 8]). This can be then generalized to FO+MOD-transductions.

▶ **Theorem 27.** *Let $\mathcal{C}$ be a set of binary structures with bounded twin-width, and* T *be an* FO+MOD*-transduction. Then $\mathsf{T}(\mathcal{C})$ has bounded twin-width.*

Proof. This follows from [8, Section 8] with the change of the previous theorem. ◀

As an illustrative example, we give a simple consequence of Theorem 27: shallow vertex minors of bounded twin-width graphs have bounded twin-width. The *local complementation* at a vertex v of a graph G consists of replacing, in G, the induced subgraph $G[N(v)]$ by its complement. A graph H is a *depth-1 vertex minor* of a graph G if there exists an independent set I of G and a subset $R \subseteq G$, such that H can be obtained from G by local complementation at the vertices of I, then deletion of the vertices in R. Note that the result does not depend on the order in which the local complementations are performed as I is an independent set. A *depth-d vertex minor of G* is inductively defined as a depth-1 vertex minor of a depth-$(d-1)$ vertex minor of G.

▶ **Proposition 28.** *For every pair of integers d, t, there is an integer t' such that every depth-d vertex minor of a graph with twin-width at most t has twin-width at most t'.*

Proof. We shall just show that the *depth-1* vertex minors of graphs with bounded twin-width have bounded twin-width. The class $\mathcal{D}_t$ of all depth-1 vertex minors of graphs with twin-width at most t is an FO+MOD-transduction of the set of all graphs with twin-width at most t.

The transduction works as follows: considering I and R as unary predicates, the new domain is defined as the set of vertices not in R, and the new adjacency relation is defined by the formula $\neg(x = y) \wedge \big(E(x,y) \leftrightarrow (\exists^e z\ I(z) \wedge E(x,z) \wedge E(y,z))\big)$, that is: two vertices u and v have their adjacency complemented if they have an odd number of common neighbors in I. ◀

We now show that the parity-minor closure of a matrix class can be expressed as an FO+MOD-transduction.

▶ **Lemma 29.** *Let $\mathbb{F}$ be a finite field. There is an* FO+MOD*-transduction T_{pm} such that, for every matrix class $\mathcal{M}$ over $\mathbb{F}$, $\mathsf{T}_{pm}(\mathcal{M}) = \mathrm{Clos}_{pm}(\mathcal{M})$.*

Proof. We first show the lemma when $\mathbb{F}$ is the binary field. The proof of that fact already contains all the ideas of the general case without presenting "unnecessary" technicalities. We recall that our 0,1-matrices M are τ-structures with $\tau = (R, \prec, E)$. The universe is the union of the sets of row and column indices, $\prec$ is interpreted as a linear order on the indices (with the row preceding the column indices), and $M \models E(x,y)$ whenever there is a 1-entry at row x, column y in M. We write $x \preccurlyeq y$ as a short-hand for $x = y \vee x \prec y$.

The transduction T_{pm} uses two (non-deterministic) unary relations U and D. We interpret U as the rows and columns kept to form the parity minor, and D as the first row (resp. column) of a row part (resp. column part). Let N be any parity minor of M. Crucially there is an even number of 1-entries in a cell of a division of M if and only if the number of rows with an odd number of 1-entries is even. This can be expressed by two nested modular counting quantifiers, and $E^N(x,y)$ can be FO+MOD-defined as

$$\mathrm{Valid}(x,y)\ \wedge\ \exists^e r\ \exists^o c\ \mathrm{Rectangle}(x,y,r,c)\ \wedge\ E^M(r,c),\ \text{with}$$

$$\mathrm{Valid}(x,y) = R(x)\ \wedge\ \neg R(y)\ \wedge\ D(x)\ \wedge\ D(y),\ \text{and}$$

$$\mathrm{Rectangle}(x,y,r,c) = R(r)\ \wedge\ \neg R(c)\ \wedge\ U(r)\ \wedge\ U(c)\ \wedge\ x \preccurlyeq r\ \wedge\ y \preccurlyeq c$$

$$\wedge\ \forall r'\ (D(r')\ \wedge\ x \prec r') \to r \prec r'\ \wedge\ \forall c'\ (D(c')\ \wedge\ y \prec c') \to c \prec c'.$$

For the sake of legibility, we omitted the M superscript for all the relations but E^M. The universe of N is defined as the indices x such that $\models D(x)$, and $\prec^N$ is naturally inherited from $\prec^M$.

One can check that there is a perfect correspondence between the parity-minor operations and the formula $E^N(x,y)$. Rows or columns x such that $\models \neg U(x)$ are removed. The lines of the division start just before every row or column x such that $\models D(x)$. All the rows or columns x such that $\models \neg\exists x'\ D(x') \ \wedge\ x' \preccurlyeq x$ could be equivalently removed. Therefore $T_{\mathrm{pm}}(\mathcal{C}) = \mathrm{Clos}_{\mathrm{pm}}(\mathcal{C})$ holds.

We now deal with the general case. Let p be the number of elements in the finite field $\mathbb{F}$. We thus consider τ_p-structures with $\tau_p = (R, \prec, E_1, E_2, \dots, E_{p-1})$, where R, $\prec$ are as before, while the relation symbols E_i all have arity 2. Our theory contains the sentence

$$\forall x,y\ \ (R(x)\ \wedge\ \neg R(y))\ \rightarrow \bigwedge_{i\in[p-1]} \left(E_i(x,y) \rightarrow \bigwedge_{j\in[p-1]\setminus\{i\}} \neg E_j(x,y) \right).$$

So there is a unique entry $i \in [0, p-1]$ at row x, column y; namely the unique $i \in [p]$ such that $\models E_i(x,y)$ if it exists, or 0 otherwise.

We need to generalize the binary counting ("$\exists^e r\ \exists^o c$") of the sum within a rectangular zone. Let us denote by $\tilde{i}$ the element of $\mathbb{F}$ corresponding to E_i, and $\tilde{0} = 0_{\mathbb{F}}$. If p is prime, then simply $\tilde{i} = i$. We can now define $E_i^N(x,y)$ as

$$\mathrm{Valid}(x,y) \wedge \bigvee_{a:[p-1]\times[0,p-1]\to[0,p-1]} \bigwedge_{\substack{j\in[p-1]\\ k\in[0,p-1]}} \exists^{a(j,k)[p]} r\ \exists^{k[p]} c\ \mathrm{Rectangle}(x,y,r,c) \wedge E_j^M(r,c).$$
$$\sum_{\substack{j\in[p-1]\\ k\in[0,p-1]}} a(j,k)k\cdot\tilde{j}=\tilde{i}$$

Let us observe that in the previous formula, "$a(j,k)k \cdot \tilde{j}$" does *not* involve any multiplication in $\mathbb{F}$. The term $a(j,k)k$ is a scalar integer, thus $a(j,k)k \cdot \tilde{j}$ is a sum of up to $(p-1)^2$ occurrences of $\tilde{j}$.

The integer $a(j,k)$ aims to match the number modulo p of rows with k modulo p occurrences of $\tilde{j}$. Since the order of every element $\tilde{j}$ in $(\mathbb{F}, +)$ divides p,

$$\bigwedge_{k\in[0,p-1]} \exists^{a(j,k)[p]} r\ \exists^{k[p]} c\ \mathrm{Rectangle}(x,y,r,c)\ \wedge\ E_j^M(r,c)$$

indeed holds when the number of $\tilde{j}$-entries in the eligible region is equal to $\sum_{k\in[0,p-1]} a(j,k)k$ modulo p. Thus, if this holds for every $j \in [p-1]$, the sum of the entries in the eligible region is $\sum_{j\in[p-1],k\in[0,p-1]} a(j,k)k\cdot\tilde{j}$, which happens to be $\tilde{i}$. ◀

Similarly we will now show that the linear-minor closure can be expressed by an FO+MOD-transduction. This is based on the proof of the previous lemma and is only slightly more technical.

▶ **Lemma 30.** *Let $\mathbb{F}$ be a finite field. There is an* FO+MOD*-transduction T_{lm} such that, for every matrix class $\mathcal{M}$ over $\mathbb{F}$, $\mathsf{T}_{lm}(\mathcal{M}) = \mathrm{Clos}_{lm}(\mathcal{M})$.*

Proof. Let p be the cardinality of $\mathbb{F}$. We still use the unary relation D to encode the division. The main difference with the proof of Lemma 29 is that T_{lm} uses p extra relations $U_0, \dots, U_{p-1}$ instead of just U. Their interpretation is that $M \models U_i(x)$ holds whenever the weighting of row or column x is $\tilde{i} \in \mathbb{F}$. We impose that the unary relations U_i partition the set of row and column indices (i.e., the $\mathbb{F}$-weighting is indeed a mapping) with the sentence

$$\forall x \bigvee_{i\in[0,p-1]} U_i(x)\ \wedge \bigwedge_{i\in[0,p-1]} \left(U_i(x) \rightarrow \bigwedge_{j\in[0,p-1]\setminus\{i\}} \neg U_j(x) \right).$$

Again R^N and $\prec^N$ are naturally inherited from R^M and $\prec^M$. We shall now define $E_i^N(x,y)$ as

$$\mathrm{Valid}(x,y) \wedge \bigvee_{\substack{a:[p-1]\times[0,p-1]^3\to[0,p-1]\\ \sum_{\substack{j\in[p-1]\\ k,g,h\in[0,p-1]}} a(j,k,g,h)k\cdot\tilde{g}\tilde{h}\tilde{j}=\tilde{i}}} \bigwedge_{\substack{j\in[p-1]\\ k,g,h\in[0,p-1]}} \exists^{a(j,k,g,h)[p]} r\ \exists^{k[p]} c\ \mathrm{Rectangle}(x,y,r,c)$$

$$\wedge\ U_g(r)\ \wedge\ U_h(c)\ \wedge\ E_j^M(r,c),$$

where $\mathrm{Valid}(x,y)$ is defined as before while $\mathrm{Rectangle}(x,y,r,c)$ is now simply

$$R(r)\wedge\neg R(c)\wedge x\preccurlyeq r\wedge y\preccurlyeq c\wedge\forall r'\,(D(r')\wedge x\prec r')\to r\prec r'\wedge\forall c'\,(D(c')\wedge y\prec c')\to c\prec c'.$$

Now $a(j,k,g,h)$ aims to match the number modulo p of rows mapped by the $\mathbb{F}$-weighting to $\tilde{g}$ with k modulo p occurrences of $\tilde{j}$ on columns mapped by the $\mathbb{F}$-weighting to $\tilde{h}$. ◀

5 Equivalence of bounded twin-width and linear-minor freeness

As a corollary of the previous section, we obtain that bounded twin-width matrix classes over finite fields are linear-minor free, and in particular parity-minor free.

▶ **Lemma 31.** *Let $\mathbb{F}$ be a finite field. For every matrix class $\mathcal{M}$ over $\mathbb{F}$ of bounded twin-width, $\mathcal{M}$ is linear-minor free.*

Proof. By Lemma 30, there is an FO+MOD-transduction T_{lm} with $T_{\mathrm{lm}}(\mathcal{M}) = \mathrm{Clos}_{\mathrm{lm}}(\mathcal{M})$. Thus by Theorem 27, $\mathrm{Clos}_{\mathrm{lm}}(\mathcal{M})$ has bounded twin-width, and cannot be the set of all $\mathbb{F}$-matrices. ◀

For the converse, we will need the notion of *rank Latin divisions* previously introduced [7]. For any integers $k \geqslant 2$ and $d \geqslant 1$, a *rank-k Latin d-division* of a $kd^2\times kd^2$ matrix M is a regular d-division $\mathcal{D}$ of M that can be refined into a regular d^2-division $((R_1,\dots,R_{d^2}),(C_1,\dots,C_{d^2}))$ such that

- $\forall i\in[d^2]$, $M[R_i,C_j]$ is constant for every $j\in[d^2]$ but one j_i for which it has rank k,
- $\forall j\in[d^2]$, $M[R_i,C_j]$ is constant for every $i\in[d^2]$ but one i_j for which it has rank k,
- and every cell of $\mathcal{D}$ contains exactly one $M[R_i,C_j]$ with rank k.

See the left-hand side of Figure 5 for an illustration. We notice that every $M[R_i,C_j]$ is a $k\times k$ matrix, since the division $\mathcal{D}$ and its refinement are assumed regular. In particular the non-constant submatrices $M[R_i,C_j]$ are full rank. The definition we give in the previous paper of the series [7] is more restrictive: The full-rank submatrices $M[R_i,C_j]$ are canonical (Ramsey-minimal) and the pattern their positions draw within the matrix M is fixed (following Figure 5). As we do not need these additional properties, we relaxed the definition here. It was previously shown that matrix classes with unbounded twin-width contain matrices with rank-k Latin d-divisions for arbitrarily large values of k and d. For our purpose we will only need $k=2$ and d diverging.

▶ **Lemma 32** ([7])**.** *Let $\mathcal{M}$ be a matrix class of unbounded twin-width over a finite field. Then for every d, there is a matrix $M\in\mathcal{M}$ with a rank-2 Latin d-division.*

Equipped with that technical lemma, the following fact roughly boils down to Figure 5.

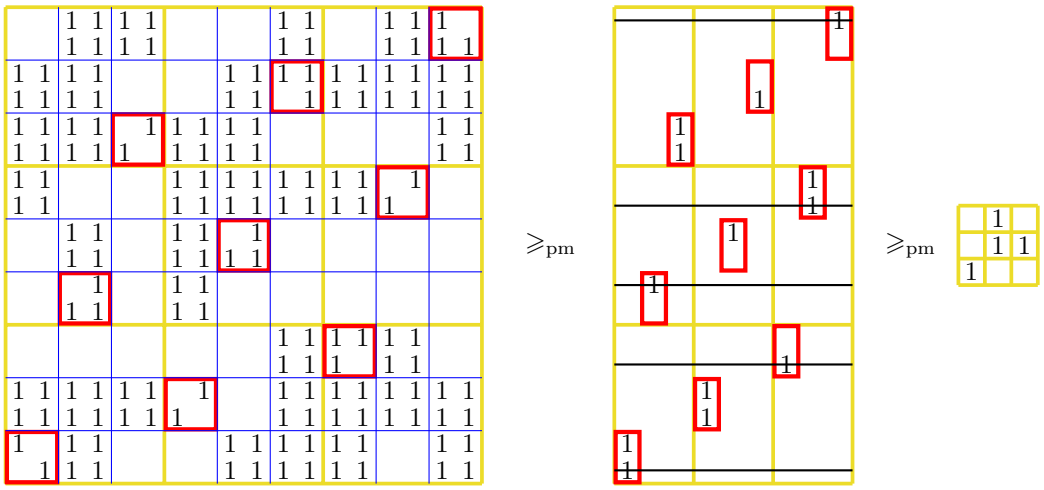

Figure 5 To the left, an example of a rank-2 Latin 3-division in $\mathbb{F}_2$. The division $\mathcal{D}$ is represented in yellow, its refinement in blue, and the rank-2 submatrices are highlighted by red boxes. Any 3×3 binary matrix, say, $\left(\begin{smallmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{smallmatrix}\right)$ can be obtained as parity minor. First sum the two columns in each vertical part of the blue division. Then correct the parity by removing at most one row per red box (middle matrix). Finally sum the cells of the yellow division (right matrix).

▶ **Lemma 33.** *Let $\mathbb{F}$ be a finite field and $\mathcal{M}$ be a matrix class of $\mathbb{F}$-matrices. If $\mathcal{M}$ is linear-minor free then it has bounded twin-width.*

Proof. We show the contrapositive. We first prove when $\mathbb{F}$ is the binary field. Let N be any $n \times n$ 0,1-matrix. By Observation 25, we want to show that N is a *parity* minor of some matrix in $\mathcal{M}$. Indeed observe that if all the square 0,1-matrices are in the parity-minor closure $\mathrm{Clos}_{\mathrm{pm}}(\mathcal{M})$, then all the 0,1-matrices are.

By Lemma 32, there is a matrix $M \in \mathcal{M}$ with a rank-2 Latin n-division $\mathcal{D}$. Let $((R_1, \ldots, R_{n^2}), (C_1, \ldots, C_{n^2}))$ be the refinement of $\mathcal{D}$ satisfying the properties of rank Latin divisions. We first sum (modulo 2) the two columns of C_j, renaming the resulting column c_j, for every $j \in [n^2]$. We call M' the obtained $2n^2 \times n^2$ matrix, and still refer to $\mathcal{D}$ for the division induced by the initial $\mathcal{D}$ on M'. For every $j \in [n^2]$, $M'_{R_{i,c_j}}$ is a 2×1 matrix equal to $\left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right)$ for every $i \in [n^2]$ but one value i_j for which it is necessarily different from $\left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right)$. Indeed summing the two columns of a 2×2 of rank 2 cannot yield a zero vector, while for both $\left(\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right)$ and $\left(\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}\right)$ it does.

By definition of a rank Latin division, the non-zero 2×1 submatrices corresponding to the initial rank-2 submatrices are on pairwise disjoint sets of rows. For each $i, j \in [n]$, in the (i, j)-cell of $\mathcal{D}$ (in M'), we can thus remove one row containing a 1 whenever the parity of the number of 1-entries within that cell does not match $N_{i,j}$. After these deletions, we sum all the remaining rows and columns in each row, then column, part of $\mathcal{D}$, and obtain the matrix N. See Figure 5 for a visual depiction.

The proof for the general case is not more complicated. Instead of summing the two columns c'_j, c''_j of C_j, we perform the weighted sum $c'_j - c''_j$ (by picking $\alpha = 1_{\mathbb{F}}$ and $\beta = -1_{\mathbb{F}}$) and again call the resulting column c_j, and the resulting matrix, M'. This has the same effect as canceling every constant submatrices, while *not* canceling every rank-2 submatrix. Now for every 2×1 submatrix on column c boxed in red in Figure 5 within the (i, j)-cell of $\mathcal{D}$, traversed by, say, rows r and r', we perform the weighted sum $(N_{i,j}x^{-1})r + 0_{\mathbb{F}}r'$ if

$M'_{r,c} = x \neq 0_{\mathbb{F}}$ (it is not possible that $M'_{r,c} = M'_{r',c} = 0_{\mathbb{F}}$). We finally get N by (simply) summing every row (or column) part of $\mathcal{D}$ into a single row (or column). ◀

We can then add to the list of characterizations of bounded twin-width for matrix classes on finite fields (see [7, Theorem 1]) a handful of new equivalent conditions.

▶ **Theorem 34.** *Given a matrix class $\mathcal{M}$ over a finite field, the following are equivalent.*

(*i*) $\mathcal{M}$ *has bounded twin-width.*
(*ii*) $\mathcal{M}$ ***is linear-minor free.***
(*iii*) $\mathcal{M}$ *is monadically dependent, i.e., for every* FO*-transduction* T, $\mathsf{T}(\mathcal{M}) \neq \mathcal{M}_{\mathrm{all}}$.
(*iv*) ***For every*** FO+MOD***-transduction*** T**,** $\mathsf{T}(\mathcal{M}) \neq \mathcal{M}_{\mathrm{all}}$**.**
(*v*) $\mathcal{M}$ *is small.*
(*vi*) ***For every*** FO+MOD***-transduction*** T**,** $\mathsf{T}(\mathcal{M})$ ***is small.***
(*vii*) $\exists n_0 \in \mathbb{N}$, $|\mathcal{M}_n| \leqslant n!$, $\forall n \geqslant n_0$.
(*viii*) *Given $M \in \mathcal{M}$ and $\varphi \in \mathrm{FO}[\tau]$, $M \models \varphi$ can be decided in $f(|\varphi|) \cdot |M|^{\mathcal{O}(1)}$.*
(*ix*) ***Given*** $M \in \mathcal{M}$ ***and*** $\varphi \in \mathrm{FO+MOD}[\tau]$**,** $M \models \varphi$ ***can be decided in*** $f(|\varphi|) \cdot |M|^{\mathcal{O}(1)}$**.**

Proof. All the items that are *not* in bold were previously proven equivalent [7]. We shall now prove that (*ii*), (*iv*), (*vi*), and (*ix*) are also equivalent to bounded twin-width. Let us note that the implication from (*viii*) (and from (*ix*)) to the other items is conditional on the complexity-theoretic assumption $\mathsf{FPT} \neq \mathsf{AW}[*]$. The equivalence (*i*) $\Leftrightarrow$ (*ii*) is what Lemmas 31 and 33 show.

(*iv*) directly implies (*iii*) since an FO-transduction is a particular FO+MOD-transduction, and (*vi*) directly implies (*v*) since the identity is an FO+MOD-transduction. For every matrix class $\mathcal{M}$ of bounded twin-width and FO+MOD-transduction T, $\mathsf{T}(\mathcal{M})$ has bounded twin-width, by Theorem 27. Thus $\mathsf{T}(\mathcal{M})$ is small and not equal to $\mathcal{M}_{\mathrm{all}}$. This means that (*i*) implies (*iv*) and (*vi*).

Finally (*ix*) immediately implies (*viii*), while the converse is given by Theorem 26. As for ordered binary structures there is a polytime algorithm to find $\mathcal{O}(1)$-sequences (when one exists) [7, Theorem 2], we do *not* need to require that the contraction sequence is provided in the input. ◀

Theorem 34 could be equivalently stated in terms of hereditary classes of ordered binary structures. Let us discuss what the new equivalent conditions of Theorem 34 (in bold) bring.

The equivalences (*viii*) $\Leftrightarrow$ (*ix*) and (*iii*) $\Leftrightarrow$ (*iv*) answer a couple of questions in the special case of ordered binary structures. It is indeed an intriguing open question whether FO+MOD model checking is strictly harder than FO model checking; more precisely, whether there is a hereditary class of graphs (or binary structures) on which FO model checking is fixed-parameter tractable while FO+MOD model checking is $\mathsf{W}[1]$-hard. Theorem 34 shows that if any such class exists, it has to be found among the *unordered* binary structures. Similarly, we do not know if the monadic dependence of a hereditary set of binary structures is equivalent to the absence of an FO+MOD-transduction onto the set of all graphs. Replacing FO+MOD-*transduction* by FO-*transduction*, this is a theorem by Baldwin and Shelah [3], which we actually took for the definition of monadic dependence.

Thus, it is again a question about the extra power (or lack thereof) of modular counting quantifiers. The equivalence (*iii*) $\Leftrightarrow$ (*iv*) settles this question when restricted to ordered binary structures. Note that there are many properties that are expressible in FO+MOD but not in FO, the simplest being that *the universe has an even number of elements* (over the empty signature). However, it is unclear if these properties can be used for a separation of the kinds (*viii*) $\not\Rightarrow$ (*ix*) or (*iii*) $\not\Rightarrow$ (*iv*) on unordered binary structures.

The equivalence $(i) \Leftrightarrow (ii)$ gives an elegant characterization of bounded twin-width for ordered binary structures. As we open our introduction with, it is reminiscent of the ties between treewidth and graph minors. As bounded treewidth can be characterized as avoiding one planar graph as a minor, bounded twin-width can be characterized for matrix classes over finite alphabets as avoiding one matrix as a linear minor. Finally $(v) \Rightarrow (vi)$ is a bit of a curiosity. It says that one cannot leave the realm of small ordered classes (starting from a hereditary class) by means of a FO+MOD-transduction.

One might dislike in the characterization $(i) \Leftrightarrow (ii)$ the seemingly arbitrary multipliers inherent to the definition of a weighted sum. By Observation 25 for $0,1$-matrices, bounded twin-width can be characterized more neatly with the mere parity minors.

▶ **Theorem 35.** *Let $\mathcal{M}$ be a matrix class over $\mathbb{F}_2$. $\mathcal{M}$ has bounded twin-width if and only if it is parity-minor free.*

The previous statement might hold more generally in any prime field. For non-prime fields, an easy counterexample is the set of all the $0, x$-matrices, where $x \neq 0_{\mathbb{F}}$ has order in $(\mathbb{F}, +)$ strictly smaller than $|\mathbb{F}|$.

6 Fixed-parameter algorithms for matrix division problems

We show how to use the results of Section 4 and the approximation algorithm of matrix twin-width [7, Theorem 2], to decide matrix problems involving a division (of a submatrix) with some FO+MOD-definable properties in fixed-parameter time. This is based on a win-win argument generalizing the algorithmic scheme of Guillemot and Marx [26] to solve Permutation Pattern, and somewhat resembling the bidimensionality technique [21]. It allows for instance to detect a k-grid minor or a k-mixed minor in an $n \times n$ matrix, or to decide if a $k \times k$ matrix is a parity or linear minor of an $n \times n$ matrix in time $f(k)n^{\mathcal{O}(1)}$.

We recall three results from the previous paper of the series. The first one is an approximation algorithm for the twin-width of matrices on finite alphabets. It outputs a large rich division if the twin-width is too high.

▶ **Theorem 36** ([7])**.** *Given as input an $n \times m$ matrix M over a finite field $\mathbb{F}$, and an integer k, there is a $2^{2^{\mathcal{O}(k^2 \log k)}}(n+m)^{\mathcal{O}(1)}$-time algorithm which returns*
- *either a $2k(k+1)$-rich division of M, certifying that $\mathrm{tww}(M) > k$,*
- *or a contraction sequence certifying that $\mathrm{tww}(M) = 2^{\mathcal{O}(k^4)}$.*

The second result is the fact that huge rich divisions contain large rank divisions.

▶ **Theorem 37** ([7])**.** *Let $\mathbb{F}$ be a finite field and K be equal to $|\mathbb{F}|^{|\mathbb{F}|^k \mathrm{mt}(k|\mathbb{F}|^k)}$. Every $\mathbb{F}$-matrix M with a K-rich division has a rank-k division.*

The third result turns rank divisions into the more structured rank Latin divisions.

▶ **Lemma 38** ([7])**.** *Let $\mathbb{F}$ be a finite field. There is a computable function $f : \mathbb{N} \to \mathbb{N}$ such that every $\mathbb{F}$-matrix with a rank-$f(k)$ division has a submatrix with a rank-k Latin division.*

We derive the following convenient corollary.

▶ **Theorem 39.** *Given as input an $n \times m$ matrix M over a fixed finite field $\mathbb{F}$, and an integer k, there is an $f(k)(n+m)^{\mathcal{O}(1)}$-time algorithm which returns*
- *either a rank-k Latin division of a submatrix of M,*
- *or a contraction sequence certifying that $\mathrm{tww}(M) \leqslant g(k)$.*

where f and g are computable functions.

Proof. This is a direct consequence of Theorems 36 and 37 and Lemma 38. The proof of Theorem 37 is effective and can readily be turned into an FPT algorithm. Indeed besides some rank computations and vector comparisons, it mainly uses the Marcus-Tardos theorem Theorem 8, which is effective (see [25, Appendix A]). The proof of Lemma 38 consists of successive Ramsey extractions, which can be done in polynomial time. ◀

A parameterized problem Π, taking as input a matrix over a fixed finite field and a non-negative integer, is FO+MOD-*definable* if, there is a computable function f, and for every non-negative integer k, there is a FO+MOD$[\tau]$ sentence $\varphi_{\Pi,k}$ of size $f(k)$ such that (M,k) is a YES-instance of Π if and only if $M \models \varphi_{\Pi,k}$. A parameterized problem Π, taking as input a matrix M and a non-negative integer k, is said *rank-bidimensional* if, for some computable functions f and g, the existence of a rank-$f(k)$ Latin division in M (i.e., a submatrix of M has a rank-$f(k)$ Latin division) permits to decide Π in time $g(k)|M|^{\mathcal{O}(1)}$. We can now state the main observation of this section.

▶ **Theorem 40.** *Every* FO+MOD-*definable rank-bidimensional problem is in* FPT.

Proof. Let Π be a rank-bidimensional problem, with computable functions f and g. Let (M,k) be an input of Π. We run the algorithm of Theorem 39 with parameter $f(k)$. In time $f'(k)|M|^{\mathcal{O}(1)}$, this either yields a rank-$f(k)$ division of a submatrix of M, and we can decide (M,k) in further time $g(k)|M|^{\mathcal{O}(1)}$ (since Π is rank-bidimensional), or a $g'(k)$-sequence of M, and we can conclude by Theorem 26 (since Π is FO+MOD-definable). ◀

As a corollary of Theorem 40, we obtain for instance that deciding if N is a parity minor of M is fixed-parameter tractable in the size of N.

▶ **Theorem 41.** *Let N be a $k \times k$ matrix, and M be an $n \times m$ matrix, both over $\mathbb{F}_2$. One can decide $N \leqslant_{pm} M$ in time $f(k)(n+m)^{\mathcal{O}(1)}$ for some computable function f.*

Proof. Let us call this problem parameterized by k, Parity Minor Containment. By the proof of Lemma 33, Parity Minor Containment is rank-bidimensional. We shall then prove that Parity Minor Containment is FO+MOD-definable, and conclude by Theorem 40. This is not a mere consequence of the proof of Lemma 29 since we can no longer rely on the non-deterministic augmentation by unary relations to only keep the desired rows and columns.

Instead, we claim that any $k \times k$ parity minor can be realized (after deletions) by a division where every row part and every column part have size at most $k+1$. Indeed, let us associate to a row or column x the parity vector $p(x) \in \mathbb{F}_2^k$ corresponding to the parity of its number of 1-entries in each of the k cells it intersects. Since $\dim(\mathbb{F}_2^k) = k$, if a part contains at least $k+2$ rows (or columns), then one can find a set of at least one and at most $k+1$ rows summing (in $\mathbb{F}_2$) to the zero vector. This non-empty set of rows can be deleted without changing the parity minor. Importantly, at most $k+1$ rows are deleted, so at least one row is remaining (so the parity minor is well formed). It is noteworthy that we actually do not need modular counting (contrary to Lemma 29). We can express $N \leqslant_{\mathrm{pm}} M$ by an FO-sentence and conclude by invoking the algorithm of [8, Section 7].

The presence of N as a parity minor can be defined by a disjunction ψ_N for every tuple $(s(1),\dots,s(k),t(1),\dots,t(k)) \in [1,k+1]^{2k}$ of the FO-sentence

$$\exists x_1^1,\dots,\exists x_{s(1)}^1,\dots,\exists x_1^k,\dots,\exists x_{s(k)}^k,\exists y_1^1,\dots,\exists y_{t(1)}^1,\dots,\exists y_1^k,\dots,\exists y_{t(k)}^k$$

$$\bigwedge_{(i,a)\neq(i',a')} x^i_a \neq x^{i'}_{a'} \ \wedge \bigwedge_{(j,b)\neq(j',b')} y^j_b \neq y^{j'}_{b'}$$

$$\wedge \bigwedge_{1\leqslant i,j\leqslant k} \mathrm{ODD}(x^i_1,\dots,x^i_{s(i)},y^j_1,\dots,y^j_{t(j)}) \leftrightarrow E^N(i,j), \text{ where}$$

$\mathrm{ODD}(x^i_1,\dots,x^i_{s(i)},y^j_1,\dots,y^j_{t(j)})$ is the formula

$$\bigvee_{\substack{S\subseteq\{x^i_1,\dots,x^i_{s(i)}\}\times\{y^j_1,\dots,y^j_{t(j)}\}\\ |S| \text{ is odd}}} \ \bigwedge_{(x^i_a,y^j_b)\in S} E^M(x^i_a,y^j_b) \ \wedge \bigwedge_{(x^i_a,y^j_b)\notin S} \neg E^M(x^i_a,y^j_b).$$

And $M \models \psi_N$ holds if and only if N is a parity minor of M. ◀

Similarly we can invoke Theorem 40 for the following problems. In the following theorem, Linear Minor Containment is the problem of deciding $N \leqslant_{\mathrm{lm}} M$, given two matrices N, M over a fixed finite field, parameterized by $|N|$.

▶ **Theorem 42.** *Linear Minor Containment, Interval Minor Containment, k-Grid Minor, k-Mixed Minor, Rank-k Division, Rank-k Latin Division, Permutation Pattern are fixed-parameter tractable.*

Proof. It is not difficult to adapt the previous proof to show that all these problems are FO-definable and rank-bidimensional. ◀

It is noteworthy that while we only know how to *approximate* efficiently the twin-width of matrices, we can efficiently compute their maximum grid or mixed minor *exactly*. We only included Permutation Pattern in the previous theorem to highlight the fact that the framework of Theorem 40 generalizes the fixed-parameter algorithm of Guillemot and Marx [26]. Their algorithm specifically targets Permutation Pattern and has of course a much better running time.

Let us insist that the problems of Theorem 42 are solved for *general* matrices (of possibly unbounded twin-width) thanks to the twin-width theory. As witnessed by the fact that the fixed-parameter algorithm of Guillemot and Marx [26] is a surprising result and a remarkable breakthrough, it is unlikely that there is an easy alternative proof of that theorem.

Theorem 39 may also be useful for W[1]-hard problems like k-Biclique [31], that are FO-definable but *not* rank-bidimensional. Atminas et al. [2] showed that k-Biclique has a fixed-parameter algorithm in the combined parameter k *plus length of the longest induced path*. In particular, this means that k-Biclique is in FPT on P_t-free graphs. We also give a fixed-parameter algorithm for k-Biclique in a subclass of structures, but we phrase our result in terms of matrices rather than graphs. The color coding technique [1] provides an FPT reduction from k-Biclique on bipartite graphs to k-Biclique (on general graphs). Thus the former problem is W[1]-hard, by the breakthrough of Lin [31]. In the language of $0,1$-matrices, this can be equivalently phrased as finding a $k \times k$ submatrix full of 1-entries.

▶ **Theorem 43.** *Let $\mathcal{M}$ be a set of $0,1$-matrices not containing every permutation matrix. Then k-Biclique on $\mathcal{M}$ is in* FPT.

Proof. Let s be the dimension of a permutation matrix *not* in $\mathcal{M}$, and let $k' := \max(k,s)$. We call the algorithm of Theorem 39 with the input matrix M and parameter k'. If this

yields a $g(k')$-sequence, we conclude with the algorithm of [8, Section 7] since k-Biclique is FO-definable:

$$\text{BICLIQUE}_k = \exists r_1, \dots, \exists r_k, \exists c_1, \dots, \exists c_k \bigwedge_{i \neq j \in [k]} \neg(r_i = r_j) \wedge \bigwedge_{i \neq j \in [k]} \neg(c_i = c_j)$$
$$\wedge \bigwedge_{i \in [k]} R(r_i) \wedge \bigwedge_{j \in [k]} \neg R(c_j) \wedge \bigwedge_{i,j \in [k]} E(r_i, c_j).$$

If instead we get a rank-k' Latin division of a submatrix of M, we observe that at least one of the constant $k' \times k'$ submatrices has to be full with 1-entries. If not, we claim that all the $k' \times k'$ permutation matrices belong to $\mathcal{M}$. Indeed, given any k'-permutation σ, one can, for each rank-k' submatrix of the rank Latin division $\mathcal{D}$, keep a pair of row and column intersecting at a 1-entry in the $(i, \sigma(i))$-cell of $\mathcal{D}$ for every $i \in [k']$, and at a 0-entry in all the other cells. Thus M contains every $k' \times k'$ permutation matrix as a submatrix, and we conclude since a matrix class is, by definition, submatrix-closed. Since $k' \geqslant s$, this contradicts the assumption on $\mathcal{M}$.

We therefore get a $k' \times k'$ submatrix full of 1-entries, and can conclude that the instance is positive since $k' \geqslant k$. ◀

Theorem 39 permits to show that finding a half-graph of height k is in FPT on any matrix class missing one permutation matrix and one complement of permutation matrix. It may for instance also be utilized to generalize the known result that k-Induced Matching is fixed-parameter tractable on bipartite $K_{t,t}$-free graphs [17]. Let k-Induced Permutation be the problem which takes a $0,1$-matrix M and a $k \times k$ permutation matrix P in input, and asks whether P is a submatrix of M. This problem also generalizes Permutation Pattern for which M would be a permutation matrix, too. It extends k-Induced Matching in bipartite graphs to the ordered setting: Not only one wants to find k mutually induced edges, but they should realize a given permutation.

▶ **Theorem 44.** *k-Induced Permutation is in* FPT *on any $K_{t,t}$-free matrix class.*

Proof. The proof follows Theorem 43 by switching the roles of bicliques and permutation patterns. More generally the problem is fixed-parameter tractable with respect to $k + t$. ◀

The following, together with the FO-definability of the problems at hand (and the FPT algorithm for FO model checking on ordered binary structures of bounded twin-width [7]), summarizes the previously drawn algorithmic consequences.

▶ **Proposition 45.** *Every $0,1$-matrix class missing either one of the following combinations has bounded twin-width:*

- *a constant-1 matrix and a permutation matrix, or*
- *a constant-0 matrix and a complement of permutation matrix, or*
- *a constant-1 triangular matrix, a permutation matrix, and a complement of a permutation matrix.*

7 Products of bounded twin-width matrices over a finite field

Perhaps somewhat disappointingly, none of the algorithmic applications of Section 6 actually requires the expressive power of FO+MOD and Theorem 26. We will now see some interesting consequences of Section 4 for matrix multiplication, that do require modular counting. First we show that if a matrix class $\mathcal{M}$ over a finite field has bounded twin-width, then so does its set of squares $\mathcal{M}^2 = \{AB : \text{A and B are two conformal matrices of } \mathcal{M}\}$.

▶ **Theorem 46.** *There is a function $f : \mathbb{N}^3 \to \mathbb{N}$ such that for every conformal matrices A and B over $\mathbb{F}_q$, The product AB (over $\mathbb{F}_q$) has twin-width at most $f(\mathrm{tww}(A), \mathrm{tww}(B), q)$.*

Proof. Since $\begin{pmatrix} 0 & A \\ B & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & A \\ B & 0 \end{pmatrix} = \begin{pmatrix} AB & 0 \\ 0 & BA \end{pmatrix}$ and

$$\mathrm{tww}\left(\begin{pmatrix} 0 & X \\ Y & 0 \end{pmatrix}\right) = \mathrm{tww}\left(\begin{pmatrix} X & 0 \\ 0 & Y \end{pmatrix}\right) = \max(\mathrm{tww}(X), \mathrm{tww}(Y), 2),$$

we shall just prove that there is a function g such that M^2 has twin-width at most $g(\mathrm{tww}(M))$, for every $n \in \mathbb{N}$ and $M \in \mathcal{M}_n(\mathbb{F}_q)$. Identifying a matrix of $\mathcal{M}_\square(\mathbb{F}_q)$ and the corresponding τ_q-structure, there is a simple FO+MOD-interpretation S such that $\mathsf{S}(M) = M^2$ for every $M \in \mathcal{M}_\square(\mathbb{F}_q)$.

Indeed one can keep the relations R and $\prec$ as in M, and express $E_i^{M^2}(x, y)$ as

$$\bigvee_{\substack{a:[q-1]^2 \to [0,q-1] \\ \sum_{j,k \in [q-1]^2} a(j,k) \cdot (\tilde{j}\tilde{k}) = \tilde{i}}} \bigwedge_{j,k \in [q-1]} \exists^{a(j,k)[q]} z \; E_j^M(x, z) \;\wedge\; E_k^M(z, y).$$

As previously, we wrote $\tilde{i}$ for the element of $\mathbb{F}_q$ corresponding to relation E_i. The expression $\tilde{j}\tilde{k}$ is a product in $\mathbb{F}_q$, while $a(j,k) \cdot (\tilde{j}\tilde{k})$ is the sum of $a(j,k)$ occurrences of $\tilde{j}\tilde{k}$. As every element of $(\mathbb{F}_q, +)$ has an order dividing q, it is enough to count the number of pairs $(\tilde{j}, \tilde{k}) = (M_{x,z}, M_{z,y})$ modulo q, which the formula does. We do not assume that $\tilde{j}\tilde{k} = \tilde{k}\tilde{j}$ (although it does hold), so our formula would also work in non-commutative rings.

We finally invoke Theorem 27 to conclude that $\mathrm{tww}(M^2)$ is bounded by a function of $\mathrm{tww}(M)$ and q. ◀

▶ **Theorem 47.** *Let q be a prime power, and d be a natural. Let A, B be two $n \times n$ matrices over $\mathbb{F}_q$, both of twin-width at most d. One can compute the product AB in time $\mathcal{O}_{d,q}(n^2 \log n)$.*

Proof. By Theorem 11, we compute an $\mathcal{O}_{d,q}(1)$-sequence for

$$\begin{pmatrix} 0 & A \\ B & 0 \end{pmatrix}, \text{ in time } \mathcal{O}_{d,q}(n^2 \log n).$$

We conclude either by turning this contraction sequence into a twin-decomposition in time $\mathcal{O}_{d,q}(n^2)$, by Theorem 16, and invoking the upcoming practical matrix squaring of Theorem 48, or by combining the FO+MOD-interpretation of Theorem 46 (and Theorem 27) with the efficient algorithm of Gajarský et al. [23] (see Theorem 21) to compute the interpretations of bounded twin-width structures. We can finally read off the top-left block AB in

$$\begin{pmatrix} 0 & A \\ B & 0 \end{pmatrix}^2,$$

in time $\mathcal{O}_{d,q}(n^2)$.

If we chose the former approach, we now have a twin-decomposition $(\mathcal{T}, \mathcal{B})$ of AB. We can initialize an $n \times n$ matrix to all 0 entries, and for each edge of $\mathcal{B}$ labeled ℓ, fill the corresponding entries with ℓ. This takes quadratic time since we access each matrix entry at most once. If we instead went with the latter approach, we shall simply make $(q-1)n^2$ constant-time queries to build AB, $q-1$ for each entry of AB. ◀

The bottleneck of Theorem 47 is to compute the contraction sequence. Should this step be improved to run in $\mathcal{O}_{d,q}(n^2)$ time, one would get an overall quadratic algorithm to multiply two matrices of bounded twin-width.

8 Efficient square computation given a twin-decomposition

For the sake of convenience, we will now use the language of graphs rather than of matrices. Recall that the *square* of a graph G is the graph G^2 with the same vertex set and whose edge set is given by:

$$E(G^2) := \{uv \ : \ \text{there is a path of length at most 2 from } u \text{ to } v \text{ in } G\}.$$

We similarly define the *modular square* of G denoted $G^{[2]}$ whose vertex set is $V(G)$ and edge set is given by:

$$E(G^{[2]}) := \{uv \ : \ |N(u) \cap N(v)| = 1 \pmod 2\}.$$

Note that if M is an adjacency matrix of G in $\mathbb{F}_2$, then the adjacency matrix of $G^{[2]}$ is exactly M^2. More generally, for every $q \geqslant 2$, if G is a complete graph with edges labeled by a function $\nu : E(G) \to \mathbb{F}_q$, we let $G^{[q]}$ be the complete graph with vertex set $V(G)$ and edges labeled by $\lambda : E(G^{[q]}) \to \mathbb{F}_q$ defined for every $uv \in E(G^{[q]})$ by:

$$\lambda(uv) := \left(\sum_{w \in V \setminus \{u,v\}} \nu(uw)\nu(wv) \right) \in \mathbb{F}_q.$$

If M is an adjacency matrix of the labeled graph G the corresponding adjacency matrix of $G^{[q]}$. For every prime power $q = p^\alpha$, we let $m(q)$ denote the cost of basic arithmetic computations in $\mathbb{F}_q$; here only addition, subtraction, and multiplication are needed. Our main result in this section is the following:

▶ **Theorem 48.** *For every prime power $q \geqslant 2$, there is an $\mathcal{O}(m(q)d^2q^{2d}n)$-time algorithm that, given a twin-decomposition $(\mathcal{T}, \mathcal{B})$ of width d of a graph G with n vertices whose edges are labeled in $\mathbb{F}_q$, outputs a twin-decomposition of width $\mathcal{O}(d^2q^d)$ of $G^{[q]}$.*

Note that in practice, if $q = p^\alpha$ with p prime, one can choose $m(q) = \mathcal{O}(\log_p(q)^2 \log(p)^2)$ (see for example [34, Table 2.8]).

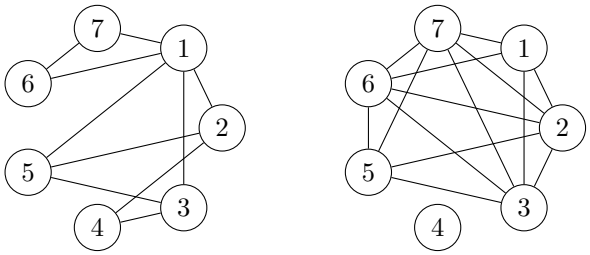

Figure 6 Left: a graph G. Right: the graph $G^{[2]}$.

Overview

We now describe a way of computing a twin-decomposition $(\mathcal{T}', \mathcal{B}')$ of $G^{[q]}$ when we we are given as input a twin-decomposition $(\mathcal{T}, \mathcal{B})$ of G. For the sake of clarity, we will only give the proof of Theorem 48 for the case $q = 2$, and later explain (at the end of this section) how to generalize it for greater values of q. From now on we assume that G is a graph of twin-width at most $d \geqslant 0$. We will first explain in Lemma 49 how to refine a d-sequence of G in order to get an $\mathcal{O}(d^2 2^d)$-sequence for $G^{[2]}$. Then we will show how to compute the

associated contraction tree $\mathcal{T}'$ from $\mathcal{T}$ using dynamic programming. The next step is to compute $\mathcal{B}'$ a set of transversal edges such that $(\mathcal{T}', \mathcal{B}')$ is a twin-decomposition of $G^{[2]}$. This is the most technical part. For this, we will first show how to construct a set of labeled edges $\mathcal{B}_1$, with labels in $\{0,1\}$, again using dynamic programming. It will follow from our construction that $\mathcal{B}_1$ have size $\mathcal{O}(d^2 4^d n)$, and that for every two vertices $u, v \in V(G)$, we can determine whether or not u and v are adjacent in $G^{[2]}$ by computing the parity of the sum of the labels over the labeled edges of $\mathcal{B}_1$ we meet between the branch corresponding to u and the one corresponding to v in $\mathcal{T}'$, plus an additional term that we will introduce later. Eventually we will prove how to compute the desired set $\mathcal{B}'$ from $\mathcal{B}_1$, using dynamic programming once more. We claim that the first two steps of our computation (i.e., the computation of $\mathcal{T}'$ and $\mathcal{B}_1$) can be done at the same time, but for the sake of clarity we will present them separately and assume that $\mathcal{T}'$ is already known when we compute $\mathcal{B}_1$.

We will adopt the following convention: We denote with lowercase characters the vertices of the original graph G and with uppercase characters the vertices of the graphs G_i and of the trees involved in the twin-decompositions, which we always identify to subsets of $V(G)$. Hence a vertex $U \in V(\mathcal{T})$ is identified to the set of leaves of the subtree of $\mathcal{T}$ rooted at U.

How to refine a d-sequence for G to a $\mathcal{O}(d^2 2^d)$-sequence for $G^{[2]}$

Assume that G admits a d-sequence $G_n, \dots, G_1$ and denote $\mathcal{P}_n, \dots, \mathcal{P}_1$ the associated partitions of $V(G)$. We define the following refinement of this sequence, denoted by $\mathcal{P}'_n, \dots, \mathcal{P}'_1$, where for each $i \in [n]$, if the red neighbors of $U \in \mathcal{P}_i$ in G_i are the sets $W_1, \dots, W_k \in \mathcal{P}_i$ (with $k \leqslant d$), then we partition U into the at most 2^{k+1} sets $(U_{(p,b_1,\dots,b_k)})_{p,b_1,\dots,b_k \in \{0,1\}}$ defined for every $p, b_1, \dots, b_k \in \{0,1\}$ by:

$$U_{(p,b_1,\dots,b_k)} := \{x \in U \ : \ \forall i \in [k], |N(x) \cap W_i| = b_i \pmod 2 \text{ and } \deg_U(x) = p \pmod 2\}.$$

In other words, we obtain $\mathcal{P}'_i$ from $\mathcal{P}_i$ by grouping together vertices from each $U \in \mathcal{P}_i$ according to the parity of their degree in U and in each of the red neighbors of U in G_i. Note that some of the sets $U_{(p,b_1,\dots,b_k)}$ may be empty. Nevertheless, at the end of the contraction sequence, $\mathcal{P}_1 = \{V(G)\}$, hence $|\mathcal{P}'_1| \leqslant 2$, as it simply corresponds to the 2-partition of $V(G)$ between vertices of even and odd degree. Thus to get a complete contraction sequence, we can eventually complete arbitrarily the sequence $\mathcal{P}'_n, \dots, \mathcal{P}'_1$ into a contraction sequence of G.

▶ **Lemma 49.** *If $\mathcal{P}_n, \dots, \mathcal{P}_1$ is the sequence of partitions of $V(G)$ associated to a d-sequence $G_n, \dots, G_1$ of G, then the refinement $\mathcal{P}'_n, \dots, \mathcal{P}'_1$ described above can be arbitrarily completed into a D-sequence for $G^{[2]}$, where $D := (d^2 + d + 1)2^{d+1} - 1$.*

Proof. We first check that the sequence $\mathcal{P}'_n, \dots, \mathcal{P}'_1$ can be completed into a contraction sequence. In other words, we show that for every $i \in [2, n]$ and for every $W_{(p,b_1,\dots,b_k)} \in \mathcal{P}'_i$, there exists some $W'_{(p',b'_1,\dots,b'_l)} \in \mathcal{P}'_{i-1}$ such that $W_{(p,b_1,\dots,b_k)} \subseteq W'_{(p',b'_1,\dots,b'_l)}$.

Assume that between G_i and G_{i-1}, the contraction described by (U, V, Z) is done, with $U, V \in \mathcal{P}_i$ and $Z := U \cup V \in \mathcal{P}'_{i-1}$. We let $W_1, \dots, W_k$ and $W'_1, \dots, W'_{k'}$ be respectively the red neighbors of U and V in G_i, with $k \leqslant d$ and $k' \leqslant d$. Observe that the red neighbors of Z in G_{i-1} are the W_js and the W'_js distinct of U and V, and possibly some other subsets of vertices from $\mathcal{P}_{i-1}$. We denote all the red neighbors of Z in G_i with $X_1, \dots, X_l$. Let $p, b_1, \dots, b_k \in \{0,1\}$. We show that there exist $p', b'_1, \dots, b'_l \in \{0,1\}$ such that $U_{(p,b_1,\dots,b_k)} \subseteq Z_{(p',b'_1,\dots,b'_l)}$. For this, for every $j \in [l]$, we let $b'_j := b_r$ if $X_j = W_r$ for some $r \in [k]$. Otherwise if $X_j \neq W_r$ for every $r \in [k]$, then it means that X_j is not a red neighbor of U in G_i, and thus we let $b'_j := 1$ if there is a complete biclique between U and X_j and

$|X_j|$ is odd, and $b'_j := 0$ otherwise. It remains to properly define p'. If UV forms a black edge in G_i, then we simply let $p' := p + |V| \pmod 2$. If UV forms a non edge in G_i, then we let $p' := p$. Finally if UV forms a red edge in G_i, then we have $V = W_j$ for some $j \in [k]$. In particular, every vertex in $U_{(p,b_1,\dots,b_k)}$ is adjacent to b_j vertices of V modulo 2. Hence every vertex of $U_{(p,b_1,\dots,b_k)}$ is adjacent modulo 2 to $p' := p + b_j \pmod 2$ other vertices of $Z = U \cup V$. By setting this value for p' we get the desired inclusion $U_{(p,b_1,\dots,b_k)} \subseteq Z_{(p',b'_1,\dots,b'_l)}$.

To prove that that the $\mathcal{P}'_i$s can be completed into a contraction sequence, it remains to show that if $W \in \mathcal{P}_i \setminus \{U, V\}$ has red neighbors $X_1, \dots, X_k$ in G_i with $k \leqslant d$ and red neighbors $Y_1, \dots, Y_l$ in G_{i-1} with $l \leqslant d$, then for every $p, b_1, \dots, b_k \in \{0,1\}$ there is $p', b'_1, \dots, b'_l \in \{0,1\}$ such that $W_{(p,b_1,\dots,b_k)} \subseteq W_{(p',b'_1,\dots,b'_l)}$. For this observe that the red neighbors of W in G_{i-1} are exactly the red neighbors of W in G_i that are distinct of U and V plus possibly the set Z. Thus for every $j \in [l]$, if $Y_j = X_i$ for some $i \in [k]$ we let $b'_j := b_i$. Moreover, we let $p' := p$. If there exists some $j \in [k]$ such that Y_j is not one of the X_is, then we must have $Y_j = Z$. Observe that by definition of $W_{(p,b_1,\dots,b_k)}$, every vertex of $W_{(p,b_1,\dots,b_k)}$ has the same number a of neighbors modulo 2 in U. Similarly, every vertex of $W_{(p,b_1,\dots,b_k)}$ has the same number b of neighbors modulo 2 in V. Hence if we let $b'_j := (a + b) \pmod 2$, we have the desired inclusion: $W_{(p,b_1,\dots,b_k)} \subseteq W_{(p',b'_1,\dots,b'_l)}$. Thus we proved that the sequence $\mathcal{P}'_n, \dots, \mathcal{P}'_1$ can be completed into a contraction sequence.

Now we need to show that this sequence is a D-contraction sequence for $G^{[2]}$. We let $G_i^{[2]}$ denote the trigraph obtained from $G^{[2]}$ when the vertex set is contracted according to the partition $\mathcal{P}'_i$. Let $i \in [n]$ and $U \in \mathcal{P}_i$. We show how to bound the red degree of every $U_{(p,b_1,\dots,b_k)}$ in $G_i^{[2]}$. Let $W_1, \dots, W_k$ be the red neighbors of U in G_i, with $k \leqslant d$. Let $p, b_1, \dots, b_k \in \{0,1\}$.

▷ Claim 50. The only red neighbors of $U_{(p,b_1,\dots,b_k)}$ in $G_i^{[2]}$ are either other subsets of U, or subsets of some sets $V \in \mathcal{P}_i$ such that the distance between U and V in the red graph of G_i is at most 2.

Proof of the claim: Let $V \in \mathcal{P}_i \setminus \{U\}$ be such that every red path from U to V in G_i has length at least 3. Let $W'_1, \dots, W'_{k'}$ be the red neighbors of V in G_i with $k' \leqslant d$, and let $p', b'_1, \dots, b'_{k'} \in \{0,1\}$. Our goal is to show that $U_{(p,b_1,\dots,b_k)} V_{(p',b'_1,\dots,b'_{k'})}$ is not a red edge in $G_i^{[2]}$. In other words, we will show that for every $u \in U_{(p,b_1,\dots,b_k)}$ and $v \in V_{(p',b'_1,\dots,b'_{k'})}$, the value $|N(u) \cap N(v)| \pmod 2$ does not depend on the choice of u and v. Observe first that by hypothesis, UV is either a black edge or a non-edge in G_i. We may assume that we are in the first case, as the second one is easier to deal with. Let $u \in U_{(p,b_1,\dots,b_k)}$ and $v \in V_{(p',b'_1,\dots,b'_{k'})}$. As $\deg_V(v) = p' \pmod 2$, we have:

$$|N(u) \cap N(v) \cap V| = p' - 1 \pmod 2.$$

By symmetry we also have:

$$|N(u) \cap N(v) \cap U| = p - 1 \pmod 2.$$

It remains to count the number of common vertices of u and v which are neither in U nor in V. There are three cases.

First if $W \in \mathcal{P}_i \setminus \{U, V\}$ is such that both UW and WV form black edges of G_i, then we have:

$$|N(u) \cap N(v) \cap W| = |W|.$$

Assume now that $W \in \mathcal{P}_i \setminus \{U, V\}$ is such that UW is a red edge in G_i and WV is a black edge in G_i. Then $W = W_j$ for some $j \in [k]$. In particular, by definition of $U_{(p,b_1,\dots,b_k)}$,

u has b_j neighbors in W modulo 2, and each of them is a neighbor of v. Thus we have:

$$|N(u) \cap N(v) \cap W_j| = b_j \pmod 2.$$

The last kind of common neighbors of u and v we can find are vertices from some $W \in \mathcal{P}_i \setminus \{U, V\}$ where UW is a black edge in G_i and WV is a red edge in G_i. By the same symmetric reasoning, we get that:

$$|N(u) \cap N(v) \cap W'_j| = b'_j \pmod 2.$$

Thus we have:

$$|N(u) \cap N(v)| = \left(p + p' + \sum_{j=1}^{k} b_j + \sum_{j=1}^{k'} b'_j \right) \pmod 2,$$

a value that does not depend on the choice of u and v. This is equivalent to say that $U_{(p,b_1,\dots,b_k)}V_{(p',b'_1,\dots,b'_{k'})}$ does not form a red edge in $G_i^{[2]}$. ◊

By Claim 50 we get that the only red neighbors of $U_{(p,b_1,\dots,b_k)}$ are either other subsets of U, or subsets of some $V \in \mathcal{P}_i \setminus \{U\}$ such that UV is a red edge in G_i, or subsets of some $V \in \mathcal{P}_i \setminus \{U\}$ such that there exists some $W \in \mathcal{P}_i \setminus \{U, V\}$ so that both UW and VW form red edges in G_i. In total this gives us at most:

$$2^{d+1} - 1 + d2^{d+1} + d^2 2^{d+1} = (d^2 + d + 1)2^{d+1} - 1$$

red neighbors of U in $G_i^{[2]}$, which is the desired result. ◀

▶ Remark 51. Observe that if we replace the condition '$|N(x) \cap W_i|$ is odd' by '$|N(x) \cap W_i| \geqslant 1$' in the definition of the refinement $\mathcal{P}'_n, \dots, \mathcal{P}'_1$, then using similar arguments than the one from the previous proof, we get a $\mathcal{O}(d^2 2^d)$-sequence for G^2.

Computation of $\mathcal{T}'$

Assume that we are given a twin-decomposition $(\mathcal{T}, \mathcal{B})$ of G, associated to a d-sequence $G_n, \dots, G_1$ of G with associated partition sequence $\mathcal{P}_n, \dots, \mathcal{P}_1$. We describe a way to compute a tree $\mathcal{T}'$ corresponding to a D-contraction sequence of $G^{[2]}$ of the form described in Lemma 49, with $D := (d^2 + d + 1)2^{d+1} - 1$. The tree $\mathcal{T}'$ will be constructed in $n - 1$ steps, and step i (for i going from $n - 1$ down to 1) is based on G_i (that is computed on the fly). We give in Figure 7 an example of the construction of $\mathcal{T}'$ for the graph of Figure 6.

We assume that $\mathcal{B}$ is lifted up. Recall this is always possible up to a preprocessing of time $\mathcal{O}(dn)$ by Remark 13. In time $\mathcal{O}(n)$, we compute for each node $U \in V(\mathcal{T})$ a parity bit $s_U \in \{0, 1\}$ such that $s_U = |U| \pmod 2$ (recall that U is naturally identified to a subset of $V(G)$).

Recall that by Lemma 15, we can dynamically compute in time $\mathcal{O}(dn)$ the lists L_U^i of the red neighborhoods of vertices $U \in V(G_i)$. Though we present it separately, the algorithm computing $\mathcal{T}'$ and the the set of partial edges $\mathcal{B}_1$ will be dynamic, and all the computations in step i are done at the same time, so we assume that at step i that we always know the lists L_U^i and the parity bits s_U.

Our goal is now to find a way to compute the refinement we described previously. For this we define for every $i \in [2, n]$, every $U \in V(G_i)$ such that $L_U^i = \{W_1, \dots, W_k\}$ and every $p, b_1, \dots, b_k \in \{0, 1\}$ the at most 2^{d+1} pointers $q^i[U, p, b_1, \dots, b_k]$ by:

$$q^i[U, p, b_1, \dots, b_k] := (V, p', b'_1, \dots, b'_l),$$

where $(v, p', b'_1, \dots, b'_l)$ is the (unique, if it exists) tuple such that $U_{(p,b_1,\dots,b_k)} \subseteq V_{(p',b'_1,\dots,b'_l)}$, with $U \in V(G_i)$ and $V \in V(G_{i-1})$. Observe that $U_{(p,b_1,\dots,b_k)}$ may be possibly empty, in which case we can set $q^i[U, p, b_1, \dots, b_k] := \emptyset$.

We show that these pointers can be dynamically computed:

▶ **Lemma 52.** *The values $q^i[U, p, b_1, \dots, b_k]$ can be dynamically computed for decreasing values of i from n down to 2. Moreover, each step takes time $\mathcal{O}(d2^d)$.*

Here, Lemma 52 has to be understood as Lemma 15: we are able to visit the nodes of $\mathcal{T}$ for decreasing values of their labels $i \in [n-1]$, and update accordingly $\mathcal{O}(n)$ pointers $q[U, p, b_1, \dots, b_k]$ such that at step i, the value of $q[U, p, b_1, \dots, b_k]$ is exactly $q^i[U, p, b_1, \dots, b_k]$ for every $U \in V(G_i)$ and $p, b_1, \dots, b_k \in \{0, 1\}$. Hence we always keep only a linear number of values $q[U, p, b_1, \dots, b_k]$, and show that, at each step, we only change a bounded number of them (namely $\mathcal{O}(d2^d)$).

Proof. As we will define q^{i-1} from q^i, we will also need to define for the initialization q^{n+1}. For this, as $G_n = G$, $\mathcal{P}_n$ is the partition of $V(G)$ into singletons and there are no red edges, we set for every $v \in V(G)$ and $p \in \{0, 1\}$:

$$q^{n+1}[\{v\}, p] := \begin{cases} (\{v\}, 0) & \text{if } p = 0 \\ \emptyset & \text{if } p = 1 \end{cases}$$

This will help us in what follows to only deal with nonempty sets $U_{(p,b_1,\dots,b_k)}$, as the empty ones will be the ones having no antecedent by q^i.

Now let $i \in [2, n]$ and assume that the contraction between G_i and G_{i-1} is (U, V, Z), with $U, V \in V(G_i)$ and $Z = U \cup V \in V(G_{i-1})$. Let $W \in V(G_i)$. We assume that $L^i_W = \{W_1, \dots, W_k\}$ is known. If $W \notin \{U, V\}$, then we distinguish four different cases. If $WU \in E(G_i)$ and $WV \in E(G_i)$ or if $WU \notin E(G_i) \cup R(G_i)$ and $WV \notin E(G_i) \cup R(G_i)$, then the adjacencies (or non-adjacencies) of W in G_{i+1} are the same as in G_i, thus for every $p, b_1, \dots, b_k \in \{0, 1\}$, we set $q^{i-1}[W, p, b_1, \dots, b_k] := q^i[W, p, b_1, \dots, b_k]$. In fact this implies that the only pointers we will update are those related to vertices of $V(G_{i-1})$ which are either linked by a red edge to U or V, or U and V themselves.

If $WU \in E(G_i)$ and $WV \notin E(G_i) \cup R(G_i)$, then we have $L^{i-1}_W = L^i_W \cup \{Z\} = \{W_1, \dots, W_k, W_{k+1} := Z\}$, so we set for every $p, b_1, \dots, b_k \in \{0, 1\}$:

$$q^i[W, p, b_1, \dots, b_k] := (W, p, b_1, \dots, b_k, s_U).$$

By hypothesis on the placement of the transversal edges of $\mathcal{B}$, we can detect this case, which happens when there is an edge of $\mathcal{B}$ between the node of $\mathcal{T}$ corresponding to U and W, and where V is neither in L^i_W, nor linked to W by an edge of $\mathcal{B}$. Of course the case where $UW \notin E(G_i)$ and $UV \in E(G_i)$ is symmetric.

If $UW \notin E(G_i) \cup R(G_i)$ and $UV \in R(G_i)$, then assume without loss of generality that $W_k = V$ and let $W'_k := Z$. $L^{i-1}_W = (L^i_W \setminus \{V\}) \cup \{Z\}$ and we set for every $p, b_1, \dots, b_k \in \{0, 1\}$:

$$q^i[W, p, b_1, \dots, b_k] := (W, p, b_1, \dots, b_k).$$

The case when $UW \notin E(G_i) \cup R(G_i)$ and $UV \in R(G_i)$ is similar, as we also have $L^{i-1}_W = (L^i_W \setminus \{W_k\}) \cup \{W'_k\}$, so if again we assume that $W_k = V$, we just need to set:

$$q^i[W, p, b_1, \dots, b_k] := (W, p, b_1, \dots, b_k + p_U).$$

The last case satisfying $W \notin \{U, V\}$ is when we have both $UW \in R(G_i)$ and $VW \in R(G_i)$. Assume for simplicity that $W_{k-1} = U$ and $W_k = V$, and set $W'_{k-1} := Z$. Then we have $L^{i-1}_W = (L^i_W \setminus \{W_{k-1}, W_k\}) \cup \{W'_{k-1}\}$ so we let:

$$q^i[W, p, b_1, \dots, b_{k-2}, b_{k-1}, b_k] := (W, p, b_1, \dots, b_{k-2}, b_{k-1} + b_k).$$

Again, all these configurations are easy to detect thanks to our assumptions on $\mathcal{B}$ and the fact we know L^i_W.

Assume now that $W = U$, and let $L^i_U = \{W_1, \dots, W_k\}$ and $p, b_1, \dots, b_k \in \{0, 1\}$. We explain now how to compute the value $q[U, p, b_1, \dots, b_k]$. Observe first that if a vertex of U has degree p in U modulo 2, then we can compute p', its degree modulo 2 in $Z = U \cup V$ by setting:

$$p' = \begin{cases} p & \text{if } UV \notin E(G_i) \cup R(G_i) \\ p + s_V & \text{if } UV \in E(G_i) \\ p + b_l & \text{if } UV \in R(G_i) \text{ and } V = W_l \end{cases}.$$

We now describe the different types of red adjacencies that can exist between Z and other vertices in G_{i-1}, and show for each of them how the associated parity bit b' can be computed. More precisely, if $u \in U_{(p,b_1,\dots,b_k)}$, we show how to compute $b' := deg_Z(u)$. Let $Y \in V(G_{i-1})$ be a red neighbor of Z in G_i. Then Y can be of five different types, and we show how to compute the red parity b' modulo 2 of the different vertices of $U \subseteq Z$ in G_{i-1} according to the one they had in G_i. First, observe that if Y was already a red neighbor of U in G_i, i.e., if $Y = W_j \in L^i_U \cap L^{i-1}_Z$ for some j, then we simply let $b' := b_j$. Assume now that $Y \in L^i_V \setminus L^i_U$. Then either there is no edge between Y and U in G_i, in which case we let $b' := 0$, or $YU \in E(G_i)$ and we let $b' := s_Y$. The last two cases are when a red edge is created between Z and Y in G_{i-1}, but Y is neither a red neighbor of U nor of V. Then again either $UY \notin E(G_i)$ and $VY \in E(G_i)$, so we let $b' := 0$, or $UY \in E(G_i)$ and $VY \notin E(G_i)$ so we let $b' := s_Y$.

We can identify all these cases as we know L^i_U, L^i_V, L^{i-1}_Z, and by the hypotheses we put on $\mathcal{B}$. Now for the complexity at step i, as the only $W \in V(G_i) \setminus \{U, V\}$ such that some $q[W, p, b_1, \dots, b_k]$ are updated are the ones such that W is a red neighbor of U or V, it means that at most $2d2^{d+1}$ pointers $q[W, p, b_1, \dots, b_k]$ are modified at step i. One can check that each of these modifications can be done in time $\mathcal{O}_d(1)$, as well as the listing of all such W. Hence we deduce the desired overall complexity for the running time at each step. ◀

Now observe by Lemmas 15 and 52 that we have everything in hand to compute $\mathcal{T}'$. Indeed, we can compute dynamically the pointers $q[U, p, b_1, \dots, b_k]$, and keep only the one associated to sets $U_{(p,b_1,\dots,b_k)}$ that are nonempty. The nodes of $\mathcal{T}'$ will be the set of every nonempty set $U_{(p,b_1,\dots,b_k)}$ we encounter. One has to take care as the set $U_{(p,b_1,\dots,b_k)}$ can be equal to some other set $U'_{(p',b'_1,\dots,b'_{k'})}$. However they will correspond to the same node of $\mathcal{T}'$. If at some step i we have $q^i[U, p, b_1, \dots, b_k] = q^i[p', b'_1, \dots, b'_{k'}] = (Z, p_Z, c_1, \dots, c_t)$, then we choose the node $Z_{(p_Z,c_1,\dots,c_t)}$ as the common parent of $U_{(p,b_1,\dots,b_k)}$ and $U'_{(p',b'_1,\dots,b'_{k'})}$. To keep a binary tree, one can replace the edges from $Z_{(p_Z,c_1,\dots,c_t)}$ to its (more than 2) children by a binary tree whose leaves are these children and root is $Z_{(p_Z,c_1,\dots,c_t)}$. This creates a bounded number of additional nodes in $\mathcal{T}'$.

In the end of the last step when $i = 1$, we may end with two disjoint rooted trees whose leaves correspond to the nodes of even degree and to the nodes of odd degree. In this case, we add an additional node that corresponds to $V(G)$ to get a rooted tree. It remains to label the nodes of $\mathcal{T}'$. By Lemma 49, one can take any total order which is consistent with the

one of the creation of the nodes in the proof (i.e., if $i < j$, we label every node associated to a set that was created in step i with a label smaller than those associated to a set created in step j).

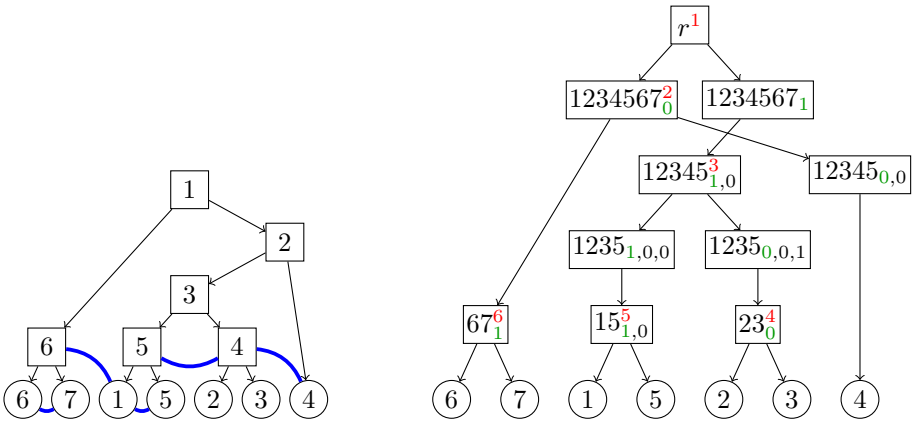

Figure 7 Left: A twin-decomposition of the graph G of Figure 6. Right: The tree $\mathcal{T}'$ associated to $(\mathcal{T}, \mathcal{B})$. We subdivided some edges of $\mathcal{T}'$ to make clear how the refinement works. The labels of the nodes of $\mathcal{T}'$ are given by the red exponents. The first bit (in green) of the tuples in subscript correspond to the bits p, and the remaining ones to the b_is.

Computation of the set of labeled transversal edges $\mathcal{B}_1$

We now compute exactly the adjacencies in $G^{[2]}$. As previously announced, we first need to compute a set $\mathcal{B}_1$ of transversal edges of $\mathcal{T}'$ with linear size in n, together with for every edge $e = UV \in \mathcal{B}_1$ a parity label $\beta_e \in \{0, 1\}$. We will ensure that $\mathcal{B}_1$ has linear size in n. Moreover we will also compute for each node $U \in V(\mathcal{T}')$ a parity bit $\alpha_U \in \{0, 1\}$. In the end, we wish to recover, with $\mathcal{B}_1$ and the bits α_U, all the adjacencies of $G^{[2]}$. More precisely, we show that for every $u, v \in V(G)$, $uv \in E(G^{[2]})$ if and only if the sum of the labels β_e over all edges $e \in \mathcal{B}_1$ between the branches of $\mathcal{T}'$ having $\{u\}$ and $\{v\}$ as children plus the sum of the parity bits α_U over all nodes U that are both predecessor of $\{u\}$ and $\{v\}$ is odd.

We again visit the nodes of $\mathcal{T}'$ by decreasing values of their labels. In fact this step can be done at the same time as the previous one (construction of $\mathcal{T}'$), so we assume that whenever the node we are visiting was created at step i of the contraction sequence of $\mathcal{T}$, we have access to the lists of red neighbors L^i_U for every $U \in V(G_i)$ and to the correspondence between the nodes of $\mathcal{T}'$ and the partition of each $U \in V(G_i)$ into $U_{(p,b_1,\dots,b_k)}$. We progressively form $\mathcal{B}_1$, starting from the empty set. In the end, if we denote with $\prec_{\mathcal{T}'}$ the partial order over the nodes of $\mathcal{T}'$ defined by $U' \prec_{\mathcal{T}'} U$ if and only if U' is an ancestor of U in $\mathcal{T}'$, then the adjacency between every two vertices u and v in $G^{[2]}$ is determined by the parity of the sum:

$$S_{uv} = \sum_{\substack{W \in V(\mathcal{T}') \\ W \prec_{\mathcal{T}'} \{u\} \\ W \prec_{\mathcal{T}'} \{v\}}} \alpha_W + \sum_{\substack{UV \in \mathcal{B}_1 \\ U \prec_{\mathcal{T}'} \{u\} \\ V \prec_{\mathcal{T}'} \{v\}}} \beta_{UV}. \tag{1}$$

See Figure 8 for an illustration of the algorithm explained below.

▶ **Lemma 53.** *One can dynamically compute in time $\mathcal{O}(d^2 4^d n)$ a set of labeled edges $\mathcal{B}_1$ of size $\mathcal{O}(d^2 4^d n)$ and parity bits α_U such that (1) holds.*

Proof. As mentioned above, we assume that we have access to the red adjacencies in $\mathcal{T}$ through the lists L^i_U, as well as the parity bits $s_U := |U| \pmod 2$ and the correspondence between the nodes of $\mathcal{T}'$ and the associated subsets in the refinement described in Lemma 49. We still assume that $\mathcal{B}$ is lifted up. Moreover, observe that it is easy to compute for every node U' of $\mathcal{T}'$ the parity of its cardinality $s'_{U'} := |U'| \pmod 2$.

Our main goal is to count (modulo 2) the number of common neighbors between every pair of distinct vertices $u, v \in V(G)$. For such a pair, we distinguish different types of common vertices, and show that we take every of them into account either with a bit α_U or with a label β_e for some $e \in \mathcal{B}_1$. We visit the nodes of $\mathcal{T}$ by decreasing order of their labels. We start with $\mathcal{B}_1 := \emptyset$ and for each node $U' \in \mathcal{T}'$, $\alpha_{U'} := 0$. The general idea is that at step i, we have a partial knowledge of G_i: we know all the red adjacencies, thanks to the lists L^i_U, and we know some of the black edges thanks to $\mathcal{B}$. At step i, we consider every black edge e of G_i that "disappears" in G_{i-1}, and compute the contribution of every path of length 2 in G containing at least one of the edges of e (recall that we can see $e \in E(G_i)$ as a set of edges of $E(G)$). Doing so, we consider each edge of G only once, which prevents us from double counting.

Assume that the i-th contraction of the d-sequence described by $\mathcal{T}$ is corresponds to the triple (U, V, Z) with $U, V \in V(G_i)$ and $Z := U \cup V \in V(G_{i-1})$. Then for every $W \in V(G_i)$ (W can possibly be V) such that $UW \in \mathcal{B}$, we do the following:

- (i) For every node $U_{(p,b_1,\dots,b_k)} \subseteq U$, we replace the value of $\alpha_{U_{(p,b_1,\dots,b_k)}}$ by $\alpha_{U_{(p,b_1,\dots,b_k)}} + s_W$. Similarly, for every node $W_{(p,b_1,\dots,b_k)} \subseteq W$ we replace $\alpha_{W_{(p,b_1,\dots,b_k)}}$ by $\alpha_{W_{(p,b_1,\dots,b_k)}} + s_U$.
- (ii) We add in $\mathcal{B}_1$ between every two distinct nodes $U_{(p,b_1,\dots,b_k)} \subseteq U$ and $U_{(p',b'_1,\dots,b'_k)} \subseteq U$ an edge e labeled by $\beta_e := s_W$. Similarly we add in $\mathcal{B}_1$ between every two distinct nodes $W_{(p,b_1,\dots,b_k)}$ and $W_{(p,b_1,\dots,b_k)}$ an edge e labeled by $\beta_e := s_U$. (Together with the previous item, this is when we count the contribution of the paths of the form $U - W - U$ or $W - U - W$.)
- (iii) We add in $\mathcal{B}_1$ between every two distinct nodes $U_{(p,b_1,\dots,b_k)} \subseteq Z$ and $W_{(p',b'_1,\dots,b'_{k'})} \subseteq W$ an edge e labeled by $\beta_e := p + p' \pmod 2$. Informally these edges count the contribution of the paths of the form $U - W - W$ or $U - U - W$.
- (iv) For every $W' \in V(G_i)$ with $W' \neq W$ such that $UW' \in \mathcal{B}$ (W' is possibly equal to V), we add in $\mathcal{B}_1$ between every two distinct nodes $W_{(p,b_1,\dots,b_k)} \subseteq W$ and $W'_{(p',b'_1,\dots,b'_{k'})} \subseteq W'$ an edge e labeled by $\beta_e := s_U$. These edges count the contribution of the paths of the form $W - U - W'$. (Together with the four previous cases, we counted so far the contribution of every 2-path that takes only black edges of G_i and "disappearing" in G_{i-1}.)
- (v) For every $X \in L^i_U$, we add in $\mathcal{B}_1$ between every two distinct nodes $X_{(p,b_1,\dots,b_k)} \subseteq X$ and $U_{(p',b'_1,\dots,b'_{k'})} \subseteq U$ an edge e labeled by $\beta_e := b_j$, where j is the index of U in L^i_X. These edges count the contribution of the paths of the form $X \sim U - W$.
- (vi) For every $X \in L^i_W$, we add in $\mathcal{B}_1$ between every two distinct nodes $U_{(p,b_1,\dots,b_k)} \subseteq U$ and $X_{(p',b'_1,\dots,b'_{k'})} \subseteq X$ an edge e labeled by $\beta_e := b'_j$, where j is the index of W in L^i_X. These edges count the contribution of the paths of the form $U - W \sim X$.

Note that in this procedure, we can add multiple labeled edges for a same pair of nodes of $\mathcal{T}'$. In this case, we replace all of them by a single edge whose label is the sum of all their labels. Observe that at step i, we add at most $(1 + (d+1) + d(d+1) + d + d)2^{2d} = \mathcal{O}(d^2 4^d)$ labeled edges (the number of candidates for W is at most $d+1$), hence $\mathcal{B}_1$ eventually has linear size in n.

We show that when the procedure terminates, the sum S_{uv} written above is odd for every pair of adjacent vertices in $G^{[2]}$, and even for the other pairs. For this, consider a pair $u, v \in V(G)$ of distinct vertices of G. Our goal is to show that every $w \in N(u) \cap N(v)$ has a contribution of exactly 1 in the sum S_{uv}. In other words, we show that every path of length 2 of the form $u - w - v$ is counted exactly once in S_{uv}. We say that the edge UV *disappears at step* i if $U, V \in V(G_i)$, $UV \in E(G_i)$ and $U'V' \notin E(G_{i-1})$, where $U', V' \in V(G_{i-1})$ are such that $U \subseteq U'$ and $V \subseteq V'$ (in particular it is the case when $U \cup V \in V(G_{i-1})$). Let $i \in [n]$, $w \in N(u) \cap N(v)$ and U, V and W be the sets of $V(G_i)$ that respectively contain u, v and w (they may be equal). Observe that for $i = n$ the edges UW and WV are in $E(G_n)$, and then disappear at some step i. Assume without loss of generality that UW disappears after VW does, and take i minimal such that $UW \in E(G_i)$. We distinguish the following disjoint cases:

- Assume first that both UW and VW disappear at step i, i.e., we also have $VW \in E(G_i)$.
  - Assume that $U = V$. As UW disappears at step i, it means that $UV \in \mathcal{B}$, either because U is merged with some other set, or because W does. T hen at step i of our algorithm, we count the paths $u - w - v$ either in (i), or in (ii) according to whether or not u and v lie in the same node of $\mathcal{T}'$.
  - If $U \neq V$, then it means that W is merged with another set in G_{i-1}. Let $W' \in V(G_{i-1})$ be the set that contains W as a subset. Then we have $UW' \in R(G_{i-1})$ and $VW' \in R(G_{i-1})$, and $UW, VW \in \mathcal{B}$. In particular this means that we counted $u - w - v$ in (iv) (with W' playing the role of Z).
- Now assume that $V = W$, i.e., that VW disappeared because VW were merged together.
  - Assume that UW disappears because U and W are merged together in G_{i-1}. Then the path $u - w - v$ is counted in (ii).
  - Assume now that U is merged with another set and contained in $U' \in V(G_{i-1})$. Then again, we claim that $u - w - v$ is counted in (ii). The case when W is merged with another set in G_{i-1} is symmetric.
- Now we assume that U, V and W are pairwise disjoint, and that $VW \in R(G_i)$.
  - Assume first that U and W are merged into $Z := U \cup W$ in G_{i-1}. Then as V is a red neighbor of W in G_i, observe that the path $u - w - v$ is counted in (vi).
  - Assume now that V and W are merged into $Z := V \cup W$ in G_{i-1}. Then as V is a red neighbor of W in G_i, observe that the path $u - w - v$ is counted in (i).
  - If instead U and V are merged into $Z := U \cup V$ in G_{i-1}. Then again, the path $u - w - v$ is counted in (vi).
  - Eventually assume that the sets containing u, v and w are still disjoint in G_{i-1}, and call them respectively U', V' and W'. Then $U'W', V'W' \in R(G_{i-1})$. Observe that as UW disappeared, we have $V' = V$ and exactly one of the two equalities $W = W'$ and $U = U'$ holds. If $U \neq U'$ and $W = W'$, then observe that the path $u - w - v$ was counted in (vi) (with U' playing the role of Z), while if $U = U'$ and $W \neq W'$, it was counted in (v).

Now it only remains to show that w is counted in only one item of our algorithm. For this, observe that there is a unique i such that the edge uw is included in a black edge $UW \in \mathcal{B}$. We are done as every path $u - w - v$ is only counted in our algorithm at the step i when UW disappears (if it disappears before VW). ◀

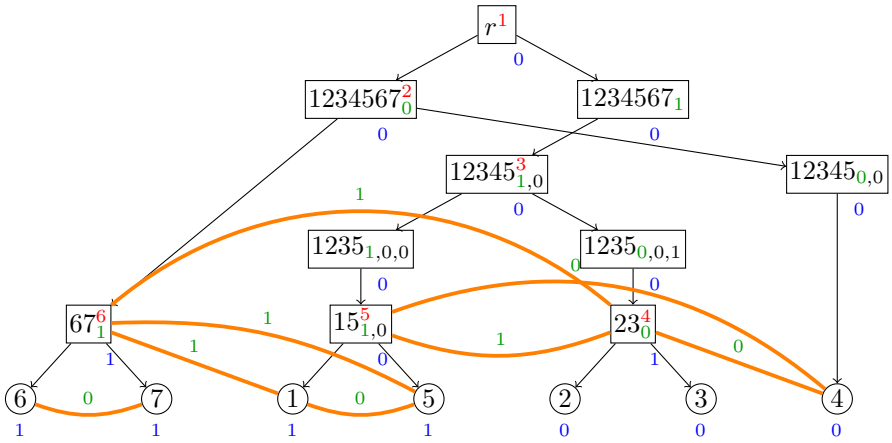

Figure 8 The result of the algorithm from Lemma 53 applied to the twin-decomposition of Figure 7. The blue bits are the values of the α_Us and the orange labeled edges represent the set $\mathcal{B}_1$. One can check that the sums S_{uv} give the adjacencies of $G^{[2]}$ pictured in Figure 6.

Computation of the set $\mathcal{B}'$ of transversal edges of $\mathcal{T}'$

In the previous sections, we built a tree $\mathcal{T}'$ together with a set of labeled $0,1$-edges $\mathcal{B}_1$ whose vertices are labeled by Booleans α_U for every $U \in V(\mathcal{T}')$, such that for every two vertices $u, v \in V(G)$, uv form an edge of $G^{[2]}$ if and only if the sum of the labels of the edges of $\mathcal{B}_1$ joining the branch having $\{u\}$ and the branch having $\{v\}$ as leaves plus the sum of the labels α_W of the internal nodes W that are common ancestors of $\{u\}$ and $\{v\}$ in $\mathcal{T}'$ is odd. We denote by $G^{[2]}_n, \dots, G^{[2]}_1$ the contraction sequence of $G^{[2]}$ that corresponds to the tree $\mathcal{T}'$.

We now compute a set of transversal edges $\mathcal{B}'$ such that $(\mathcal{T}', \mathcal{B}')$ is a twin-decomposition of $G^{[2]}$. We proceed in two steps.

(1) First, we visit the nodes of $\mathcal{T}'$ by decreasing values of their labels (i.e., from bottom to top), and try to lift up the highest we can the information they bear. This way, we modify $\mathcal{B}_1$, by removing the edges which were "too high" in $\mathcal{T}'$ to be lifted up, and construct little by little another set of labeled transversal edges $\mathcal{B}_2$ of $\mathcal{T}'$ corresponding to the edges we "blocked", i.e., that could not be pushed. We will prove that these edges correspond exactly to the black and non-edges of $G^{[2]}_i$ that disappear in $G^{[2]}_{i-1}$. Thus they correspond to the only possible place for the edges of $\mathcal{B}'$. It remains to decide their status in $G^{[2]}_i$ (i.e., whether they correspond to edges or non-edges in $G^{[2]}$).

(2) In the last step, we visit the nodes of $\mathcal{T}'$ this time by increasing values of their labels (i.e., from top to bottom). The goal is to communicate the contribution of the labels α_U and of the edges that were not removed yet from $\mathcal{B}_1$ to the lower strata of the tree $\mathcal{T}'$, in order to decide the status of each edge of $\mathcal{B}_2$.

Again we will only add or remove a constant number of edges from $\mathcal{B}_1$ and $\mathcal{B}_2$ at each step of (1) and (2), which ensures that both these sets always have linear size in n, and that our total computation time is also linear.

▶ **Lemma 54.** *One can dynamically compute bits α'_U for each $U \in \mathcal{T}'$ together with a set $\mathcal{B}_2$ of labeled edges and remove edges of $\mathcal{B}_1$ such that we eventually have the following properties:*

- *For every pair of distinct vertices $u, v \in V(G)$, the sum S_{uv} is equal to the sum S'_{uv}*

defined by:

$$S'_{uv} := \sum_{\substack{W \in V(\mathcal{T}') \\ W \prec_{\mathcal{T}'} \{u\} \\ W \prec_{\mathcal{T}'} \{v\}}} \alpha_W + \sum_{\substack{UV \in \mathcal{B}_1 \\ U \prec_{\mathcal{T}'} \{u\} \\ V \prec_{\mathcal{T}'} \{v\}}} \beta_{UV} + \sum_{\substack{UV \in \mathcal{B}_2 \\ U \prec_{\mathcal{T}'} \{u\} \\ V \prec_{\mathcal{T}'} \{v\}}} \beta_{UV}.$$

- *For every edge $e = UV \in \mathcal{B}_2$ between $U, V \in V(G_i)$, $UV \notin R(G_i^{[2]})$. Moreover, at step $i-1$, either U and V are merged together, or if U' and V' denote the disjoint sets of $V(G_{i-1})$ that respectively include U and V, we have $U'V' \in R(G_{i-1}^{[2]})$.*
- *For every remaining edge $e = UV \in \mathcal{B}_1$, we must have $UV \in R(G_i^{[2]})$.*
- *For every $U, V, Z \in V(\mathcal{T}')$ such that U and V are the two children of Z in $\mathcal{T}'$, if $\ell(Z) = i-1$ then $\alpha'_Z = 1$ if and only if $UV \in R(G_i)$.*

Furthermore, in the end, $\mathcal{B}_1$ and $\mathcal{B}_2$ have size at most $\mathcal{O}(d^2 4^d n)$.

Proof. We will keep, at step i, $n - i + 1$ lists J_U^i for each node $U \in \mathcal{B}_i(\mathcal{T}') = V(G_i^{[2]})$, updated in time $\mathcal{O}(1)$ at each step. We will see that the list J_U^i corresponds exactly to the red neighborhood of U in $G_i^{[2]}$.

We start ($i = n$) with $J_{\{v\}}^i := \emptyset$ for each $u \in V(G)$ and $\alpha'_U := 0$ for every $U \in V(\mathcal{T}')$. Now let $i \in [n]$ and assume that the contraction between $G_i^{[2]}$ and $G_{i-1}^{[2]}$ is described by the triple (U, V, Z) where $U, V \in V(G_i^{[2]})$ and $Z \in V(G_{i-1}^{[2]})$. We also assume that the lists J_U^i were already computed for every $U \in V(G_i^{[2]})$. Again, we keep the convention for $\mathcal{B}_2$ that if at some point, there are multiple edges of $\mathcal{B}_2$ between two nodes $U, V \in V(\mathcal{T}')$, then we replace them by a single edge whose label is the sum modulo 2 of all their labels. At step i of our algorithm we do the following: For every $W \in V(G^{[2]}) \setminus \{U, V\}$, we do not touch to the list J_W^i. We create the list $J_Z^i := J_U^i \cup J_V^i \setminus \{U, V\}$. These lists may be modified in what follows.

(*i*) If $V \notin J_U^i$, and if $e = UV \in \mathcal{B}_1$ with label β_e, then we remove e from $\mathcal{B}_1$ and add it in $\mathcal{B}_2$. Similarly, if $V \notin J_U^i$ and there is no edge between U and V in $\mathcal{B}_1$, we add an edge $e = UV$ labeled with $\beta_e := 0$ in $\mathcal{B}_2$. Intuitively, we mark there the position of every edge or non-edge of $G_i^{[2]}$ that disappears in $G_{i-1}^{[2]}$ because its two endpoints are merged together.

(*ii*) For every $W \in V(G_i^{[2]}) \setminus \{U, V\}$ such that $W \notin J_U^i \cup J_V^i$, if $e_1 := UW$ and $e_2 := VW$ are in $\mathcal{B}_1$ and $\beta_{e_1} = \beta_{e_2}$, then we remove e_1 and e_2 from $\mathcal{B}_1$ and add the edge $e := ZW$ in $\mathcal{B}_2$ with label $\beta_e := \beta_{e_1}$. Similarly, if $W \notin J_U^i \cup J_V^i$ and only one of the edges between e_1 and e_2 is in $\mathcal{B}_1$, say e_1 with label $\beta_{e_1} = 0$, then we remove it from $\mathcal{B}_1$ and add in $\mathcal{B}_2$ the edge $e := ZW$ with label $\beta_e := 0$. Intuitively this corresponds to the case when the partial information we kept until now of the relations between U and W and between V and W is the same, so we "lift it up" in $\mathcal{T}'$.

(*iii*) For every $W \in V(G_i^{[2]}) \setminus \{U, V\}$ such that $W \notin J_U^i \cup J_V^i$, $e_1 = UW \in \mathcal{B}_1$ and $e_2 = VW \in \mathcal{B}_1$ have different labels $\beta_{e_1} \neq \beta_{e_2}$, we remove e_1 and e_2 from $\mathcal{B}_1$ and add them in $\mathcal{B}_2$. Moreover we add W to the list J_Z^{i-1} and reciprocally we add Z to the list J_W^{i-1}. Similarly if we have $W \notin J_U^i \cup J_V^i$ and if $e_1 = UW \in \mathcal{B}_1$ but there is no edge between V and W in $\mathcal{B}_1$, then we remove e_1 from $\mathcal{B}_1$ and add in $\mathcal{B}_2$ the edges e_1 and $e_2 := VW$ with label $\beta_{e_2} := 0$. We also apply the same changes to J_W^{i-1} and J_Z^{i-1}. Intuitively this is where we mark the position of every edge or non-edge of $G_i^{[2]}$ that disappears to create a red edge where there was no error before.

(*iv*) For every $W \in V(G_i^{[2]}) \setminus \{U, V\}$ such that $W \in J_U^i$ and $W \notin J_V^i$, if the edge $e = VW$ is in $\mathcal{B}_1$, then we remove it from $\mathcal{B}_1$ and add it in $\mathcal{B}_2$. Similarly if there is no edge in $\mathcal{B}_1$ between V and W, we add in $\mathcal{B}_2$ the edge $e := VW$ labeled by $\beta_e := 0$. In both cases,

we also add Z in the list J_W^{i-1}. Intuitively this is where we mark the position of every edge or non-edge of $G_i^{[2]}$ that disappears because it is merged with some previous red edge of $G_i^{[2]}$.

(v) If $U \in J_V^i$ then we set $\alpha'_Z := 1$.

That for every $u, v \in V(G), S_{uv} = S'_{uv}$ is immediate, as we only moved edges from $\mathcal{B}_1$ to $\mathcal{B}_2$, and merged some of the edges of $\mathcal{B}_1$ in a way that does not change the value of S_{uv}.

However, it is not clear yet that the lists J_U^i have bounded size at each step. We show that indeed correspond to the red neighborhood of each vertex of $G_i^{[2]}$.

▷ Claim 55. The list J_U^i is exactly the red neighborhood of U in $G_i^{[2]}$.

Proof of the claim: We show the claim by induction over decreasing values of $i \in [n]$. If $i = n$, then the claim is immediate, as there is no red edge in the partition of $G^{[2]}$ into singletons, and we initialized the lists J_U^i to be empty.

Assume now that the claim holds for some $i \in [2, n]$, and that the contraction between $G_i^{[2]}$ and $G_{i-1}^{[2]}$ is described by the triple (U, V, Z) with $U, V \in V(G_i^{[2]})$ and $Z \in V(G_{i-1}^{[2]})$. Then a vertex $W \in V(G_i^{[2]}) \setminus \{U, V\}$ is a red neighbor of Z in $G_{i-1}^{[2]}$ if and only if it was already a red neighbor of U or V in $G_i^{[2]}$, or if it has not the same relation with U and V in $G_i^{[2]}$. In the first case, Z is indeed in J_W^{i-1} thanks to (iv). Assume now without loss of generality that $UW \in E(G_i^{[2]})$ while $VW \notin E(G_i^{[2]})$. By Lemma 53, this means that if we take any $u \in U, v \in V, W \in W$, we have: $1 = S'_{uw} \neq S'_{vw} = 0$. Now observe that every common ancestor of u and w in $\mathcal{T}'$ is also a common ancestor of v and w, and vice versa. Hence we must have:

$$\sum_{\substack{U'W' \in \mathcal{B}_1 \\ U \prec U' \prec_{\mathcal{T}'} \{u\} \\ W' \prec W' \prec_{\mathcal{T}'} \{w\}}} \beta_{U'W'} + \sum_{\substack{U'W' \in \mathcal{B}_2 \\ U \prec U' \prec_{\mathcal{T}'} \{u\} \\ W' \prec W' \prec_{\mathcal{T}'} \{w\}}} \beta_{U'W'} \neq \sum_{\substack{V'W' \in \mathcal{B}_1 \\ V \prec V' \prec_{\mathcal{T}'} \{v\} \\ W' \prec W' \prec_{\mathcal{T}'} \{w\}}} \beta_{V'W'} + \sum_{\substack{U'W' \in \mathcal{B}_2 \\ V \prec V' \prec_{\mathcal{T}'} \{v\} \\ W' \prec W' \prec_{\mathcal{T}'} \{w\}}} \beta_{V'W'}.$$

It is easy to see in the algorithm that as $W \notin J_U^i$, then there is no edge of $\mathcal{B}_1$ between the subtree of $\mathcal{T}'$ rooted in W and the one rooted in U. The same holds between W and V. Hence in the above inequality, the sums over edges of $\mathcal{B}_1$ vanish. One can also observe that the only edge in $\mathcal{B}_2$ that may exist between the subtree rooted in W and the one rooted in U is $e_1 := UW$. The same way, the only edge in $\mathcal{B}_2$ between the subtree rooted in W and the one rooted in V is $e_1 := VW$. Hence at least one of these edges must have been created in the algorithm. It could not be in (i) or (iv), as we are not in these cases, so it must have been in (iv). Thus (iii) was executed according to W, so the lists J_Z^{i-1} and J_W^{i-1} have been correctly updated. ◊

By Claim 55, we observe that the creation of an edge in $\mathcal{B}_2$ corresponds exactly to the disappearance of the edges or non-edges of $G_i^{[2]}$, which proves the second item of the lemma. Moreover, observe that in the algorithm, every edge of $e = UV \in \mathcal{B}_1$ such that $V \notin J_U^i$ is removed at one point from $\mathcal{B}_1$, so by Claim 55 we are also done with the third item of the lemma.

Now by Claim 55 and the fact that $\mathcal{T}'$ represents a contraction sequence as described in Lemma 49, we get that the size of each L_U^i is $\mathcal{O}(d^2 2^d)$. Hence, each vertex $U \in V(G^{[i]})$ has at most $\mathcal{O}(d^2 2^d)$ edges in $\mathcal{B}_2$ linking it to another node of $\mathcal{T}'$ with a higher label. Hence at each step, $|\mathcal{B}_1| + |\mathcal{B}_2|$ increases by at most $\mathcal{O}(d^2 2^d)$, and eventually $|\mathcal{B}_1| + |\mathcal{B}_2|$ has the same order than at the start, i.e., $|\mathcal{B}_1| + |\mathcal{B}_2| = \mathcal{O}(d^2 4^d n)$. The same way, we get that the execution time at each step of our algorithm is $\mathcal{O}(d^2 2^d)$. So the total running time of the procedure is $\mathcal{O}(d^2 2^d n)$. ◀

Now we have everything in hand to conclude. The set $\mathcal{B}_2$ indicates the locations of the edges and non-edges of the graphs G_i when they "disappear." It only remains to give them the appropriate label: 1 if it corresponds to an edge, and 0 is it corresponds to a non-edge. By construction, no edge of $\mathcal{B}_1$ are placed "below" an edge of $\mathcal{B}_2$, so we only need to communicate the information of the edges and the bits of the nodes that are "above" them.

▶ **Lemma 56.** *Given $\mathcal{T}', \mathcal{B}_1, \mathcal{B}_2$ computed as before, we can construct in time $\mathcal{O}(d^2 4^d n)$ a set $\mathcal{B}'$ of size $\mathcal{O}(d^2 2^d n)$ such that $(\mathcal{T}', \mathcal{B}')$ is a twin-decomposition of $G^{[2]}$ associated to the contraction sequence described in Lemma 49.*

Proof. We want each node to communicate its parity bit α_U to its children, and to transfer similarly the information of the edges that are still in $\mathcal{B}_1$ to the edges of $\mathcal{B}_2$ placed below them in $\mathcal{T}'$.

Hence we now visit the nodes of $\mathcal{T}'$ by *increasing* values of their labels (i.e., we consider the associated contraction sequence of $G^{[2]}$ starting at its end). Assume that the contraction done between $G_i^{[2]}$ and $G_{i-1}^{[2]}$ is described by the triple (U, V, Z) where $U, V \in V(G_i^{[2]})$ and $Z \in V(G_{i-1}^{[2]})$. Again, if at some point there are multiple edges of $\mathcal{B}_1$ between two nodes, we just replace them by a single one and sum their labels. At step i we do the following:

- (i) Increment the values α_U and α_V by α_Z.
- (ii) If $e = UV \in \mathcal{B}_2$, then increment its label by α_Z. If its new label is 0, remove it from $\mathcal{B}_2$. Otherwise keep it in $\mathcal{B}_2$.
- (iii) If $e = UV \notin \mathcal{B}_2$ and if $\alpha'_Z = 0$, then add in $\mathcal{B}_2$ the edge $e := UV$ whose label is $\beta_e := \alpha_Z$.
- (iv) If $e = UV \notin \mathcal{B}_2$ and $\alpha'_Z = 1$, then add in $\mathcal{B}_1$ the edge $e := UV$ whose label is $\beta_e := \alpha_Z$.
- (v) For every $W \in V(G_{i-1}) \setminus Z$ such that $e = ZW \in \mathcal{B}_1$, if $e' = UW \in \mathcal{B}_2$, increment the label of e' by β_e. Otherwise if $UW \notin \mathcal{B}_2$, add the edge $e' := UW$ in $\mathcal{B}_1$ with label $\beta_{e'} := \beta_e$. Do the same according to V. Finally remove e from $\mathcal{B}_1$.

At step i, we keep the property that if there is an edge $e = UV \in \mathcal{B}_1$ between $U, V \in V(G_i^{[2]})$, then $UV \in R(G_i^{[2]})$. Thus we ensure that every edge $\mathcal{B}_1$ will "encounter enough edges of $\mathcal{B}_2$ below it to make it disappear". Hence when the algorithm terminates, we have $\mathcal{B}_1 = \emptyset$. Moreover as the red degrees in $G_i^{[2]}$ are bounded by $\mathcal{O}(d^2 2^d)$, we also have at each step $|\mathcal{B}_1| = \mathcal{O}(d^2 2^d n)$. Similarly the total running time is $\mathcal{O}(d^2 2^d n)$.

Now observe that for every two distinct vertices $u, v \in V(G)$, the sum S'_{uv} does not change at each step. Hence, in the end, this sum is 1 if and only if the (unique) edge e of $\mathcal{B}_2$ that connects an ancestor of $\{u\}$ with an ancestor of $\{v\}$ in $\mathcal{T}'$ has label 1. If we set $:\mathcal{B}' := \{e \in \mathcal{B}_2, \beta_e = 1\}$, then by the first item of Lemma 54, $(\mathcal{T}', \mathcal{B}')$ is indeed a twin-decomposition of $G^{[2]}$. ◀

Now if we put together Lemmas 15, 49, 52–54, and 56 we get the following result which is exactly Theorem 48 when $q = 2$:

▶ **Theorem 57.** *There exists a $\mathcal{O}(d^2 4^d n)$-time algorithm that, given a twin-decomposition $(\mathcal{T}, \mathcal{B})$ of width d of a graph G with n vertices, outputs a twin-decomposition of width $\mathcal{O}(d^2 2^d)$ of $G^{[2]}$.*

▶ Remark 58. The previous algorithm can also be adapted to compute a twin-decomposition of G^2 associated to the refinement of Remark 51 in the same complexity.

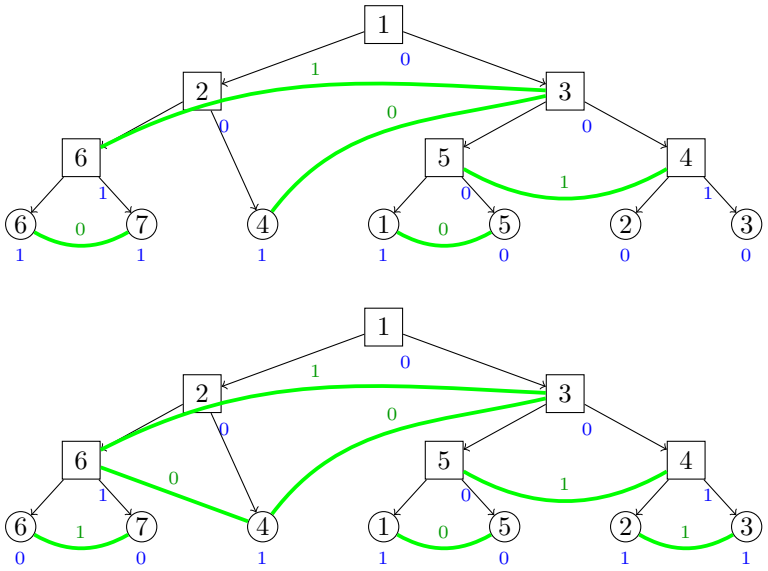

Figure 9 Top: The result of the algorithm from Lemma 54 applied to the couple $(\mathcal{T}', \mathcal{B}_1)$ from Figure 8. The green edges represent the set $\mathcal{B}_2$. For the sake of clarity, we did not add the bits α'_U. Bottom: The result of the algorithm from Lemma 56 applied to the triple $(\mathcal{T}', \mathcal{B}_1, \mathcal{B}_2)$ to the left.

The algorithm for general values of $q \geqslant 2$

As mentioned in the overview of the previous proof, there are only few changes to do in order to prove Theorem 48 in its full generality. We now explain which parts have to be modified.

We let G be a graph with edges labeled by a function $\nu : E(G) \to \mathbb{F}_q$. For every $l \in \mathbb{F}_q$ we let G_l be the graph with vertex set $V(G)$ and edge set $\nu^{-1}(l)$. For every $u \in V(G)$ and $U \subseteq V(G)$, we let $\deg_U^{(l)}(u) := |\{v \in U \ : \ \nu(uv) = l\}|$. Again let $(\mathcal{T}, \mathcal{B})$ be a d-sequence for G, with associated partition sequence $\mathcal{P}_n, \dots, \mathcal{P}_1$. The twin-decomposition $(\mathcal{T}', \mathcal{B}')$ we want to compute corresponds to the refinement of $\mathcal{P}_n, \dots, \mathcal{P}_1$ where for every $i \in [n], U \in \mathcal{P}_i$, if $W_1, \dots, W_k$ denote the at most d red neighbors of U in G_i, then $\mathcal{P}'_i$ contains all the following subsets of U which are nonempty:

$$U_{(p,b_1,\dots,b_k)} := \{x \in U \ : \ \forall i \in [k], \sum_{l \in \mathbb{F}_q} \deg_{W_i}^{(l)}(x) \cdot l = b_i$$

$$\text{and } \sum_{l \in \mathbb{F}_q} \deg_U^{(l)}(x) \cdot l = p\},$$

for every value of $(p, b_1, \dots, b_k) \in \mathbb{F}_q^{k+1}$, where for each $n \in \mathbb{N}$ and each $l \in \mathbb{F}_q$ we let $n \cdot l$ denote the sum of l with itself n times (which is not the same thing that the multiplication in $\mathbb{F}_q$). In particular, note that for each $n \in \mathbb{N}$ and $l \in \mathbb{F}_q$, $n \cdot l = (n \pmod q) \cdot l$. As previously, note that each set U is partitioned into at most q^{d+1} subsets $U_{(p,b_1,\dots,b_k)}$. The reader can check that the proof of Lemma 49 generalizes, i.e., that we can complete arbitrarily this refinement and obtain a D-sequence for $G^{[q]}$ with $D := (d^2 + d + 1)q^{d+1} - 1$. We claim that the proof of Lemma 52 also generalizes, with the additional factor $m(q)$ in the time complexity, as one needs to make a (constant) number of elementary computations in $\mathbb{F}_q$ at each update of the pointers $q[U, p, b_1, \dots, b_k]$. The only change is that we must pre-compute the values $s_U := |U| \pmod q$ for each $U \in V(\mathcal{T})$.

We let $\lambda : E(G^{[q]}) \to \mathbb{F}_q$ denote the labeling function of the edges of $G^{[q]}$. We still need to compute some values $\alpha_U \in \mathbb{F}_q$ for $U \in V(\mathcal{T}')$ and a set of labeled transversal edges $\mathcal{B}_1$ of $\mathcal{T}'$, where for each $e \in \mathcal{B}_1$, $\beta_e \in \mathbb{F}_q$ denotes the label of e such that for every $u, v \in V(G^{[q]})$:

$$\lambda(uv) = \sum_{\substack{W \in V(\mathcal{T}') \\ W \prec_{\mathcal{T}'} \{u\} \\ W \prec_{\mathcal{T}'} \{v\}}} \alpha_W + \sum_{\substack{UV \in \mathcal{B}_1 \\ U \prec_{\mathcal{T}'} \{u\} \\ V \prec_{\mathcal{T}'} \{v\}}} \beta_{UV}.$$

The algorithm described in Lemma 53 can be generalized as desired, and we get a set $\mathcal{B}_1$ as desired of size $\mathcal{O}(d^2q^{2d}n)$. Its running time becomes $\mathcal{O}(m(q)d^2q^{2d})$. Eventually, Lemmas 54 and 56 generalize in a straightforward way, up to a multiplicative cost $m(q)$ for the time complexity. Hence the overall complexity is $\mathcal{O}(m(q)d^2q^{2d}n)$ so we are done with Theorem 48.

References

1 Noga Alon, Raphael Yuster, and Uri Zwick. Color coding. In *Encyclopedia of Algorithms*, pages 335–338. 2016. doi:10.1007/978-1-4939-2864-4_76.

2 Aistis Atminas, Vadim V. Lozin, and Igor Razgon. Linear time algorithm for computing a small biclique in graphs without long induced paths. In Fedor V. Fomin and Petteri Kaski, editors, *Algorithm Theory - SWAT 2012 - 13th Scandinavian Symposium and Workshops, Helsinki, Finland, July 4-6, 2012. Proceedings*, volume 7357 of *Lecture Notes in Computer Science*, pages 142–152. Springer, 2012. doi:10.1007/978-3-642-31155-0_13.

3 John T. Baldwin and Saharon Shelah. Second-order quantifiers and the complexity of theories. *Notre Dame Journal of Formal Logic*, 26(3):229–303, 1985.

4 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)*, pages 1977–1996, 2021. doi:10.1137/1.9781611976465.118.

5 Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width III: max independent set, min dominating set, and coloring. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 35:1–35:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.35.

6 Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Pierre Simon, Stéphan Thomassé, and Szymon Torunczyk. Twin-width IV: ordered graphs and matrices. *CoRR*, abs/2102.03117, 2021. URL: https://arxiv.org/abs/2102.03117, arXiv:2102.03117.

7 Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Pierre Simon, Stéphan Thomassé, and Szymon Torunczyk. Twin-width IV: ordered graphs and matrices. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 924–937. ACM, 2022. doi:10.1145/3519935.3520037.

8 Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width I: tractable FO model checking. *J. ACM*, 69(1):3:1–3:46, 2022. doi:10.1145/3486655.

9 Édouard Bonnet, Jaroslav Nesetril, Patrice Ossona de Mendez, Sebastian Siebertz, and Stéphan Thomassé. Twin-width and permutations. *CoRR*, abs/2102.06880, 2021. URL: https://arxiv.org/abs/2102.06880, arXiv:2102.06880.

10 Steffen Börm, Lars Grasedyck, and Wolfgang Hackbusch. Introduction to hierarchical matrices with applications. *Engineering analysis with boundary elements*, 27(5):405–422, 2003.

11 S. Braunfeld and M.C. Laskowski. Existential characterizations of monadic NIP, 2022. URL: https://arxiv.org/abs/2209.05120, doi:10.48550/arXiv.2209.05120.

12 Julia Chuzhoy and Zihan Tan. Towards tight(er) bounds for the excluded grid theorem. *J. Comb. Theory, Ser. B*, 146:219–265, 2021. doi:10.1016/j.jctb.2020.09.010.

13 Josef Cibulka and Jan Kyncl. Füredi-Hajnal limits are typically subexponential. *CoRR*, abs/1607.07491, 2016. URL: http://arxiv.org/abs/1607.07491, arXiv:1607.07491.

14 B. Courcelle. Graph rewriting: an algebraic and logic approach. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume 2, chapter 5, pages 142–193. Elsevier, Amsterdam, 1990.

15 Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000. doi:10.1007/s002249910009.

16 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 4. Springer, 2015.

17 Konrad K. Dabrowski, Marc Demange, and Vadim V. Lozin. New results on maximum induced matchings in bipartite graphs and beyond. *Theor. Comput. Sci.*, 478:33–40, 2013. doi:10.1016/j.tcs.2013.01.027.

18 Sally Dong, Yin Tat Lee, and Guanghao Ye. A nearly-linear time algorithm for linear programs with small treewidth: a multiscale representation of robust central path. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1784–1797. ACM, 2021. doi:10.1145/3406325.3451056.

19 Rodney G. Downey, Michael R. Fellows, and Udayan Taylor. The parameterized complexity of relational database queries and an improved characterization of W[1]. In Douglas S. Bridges, Cristian S. Calude, Jeremy Gibbons, Steve Reeves, and Ian H. Witten, editors, *First Conference of the Centre for Discrete Mathematics and Theoretical Computer Science, DMTCS 1996, Auckland, New Zealand, December, 9-13, 1996*, pages 194–213. Springer-Verlag, Singapore, 1996.

20 Yuli Eidelman and Israel Gohberg. On a new class of structured matrices. *Integral Equations and Operator Theory*, 34(3):293–324, 1999.

21 Fedor V. Fomin, Erik D. Demaine, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Bidimensionality. In *Encyclopedia of Algorithms*, pages 203–207. 2016. doi:10.1007/978-1-4939-2864-4_47.

22 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, Michal Pilipczuk, and Marcin Wrochna. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. *ACM Trans. Algorithms*, 14(3):34:1–34:45, 2018. doi:10.1145/3186898.

23 Jakub Gajarský, Michal Pilipczuk, Wojciech Przybyszewski, and Szymon Torunczyk. Twin-width and types. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 123:1–123:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.123.

24 Erich Grädel, Phokion G Kolaitis, Leonid Libkin, Maarten Marx, Joel Spencer, Moshe Y Vardi, Yde Venema, Scott Weinstein, et al. *Finite Model Theory and its applications*. Springer, 2007.

25 Sylvain Guillemot and Dániel Marx. Finding small patterns in permutations in linear time. *CoRR*, abs/1307.3073, 2013. URL: http://arxiv.org/abs/1307.3073, arXiv:1307.3073.

26 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*, pages 82–101, 2014. doi:10.1137/1.9781611973402.7.

27 Wolfgang Hackbusch. A sparse matrix arithmetic based on H-matrices. part I: introduction to h-matrices. *Computing*, 62(2):89–108, 1999. doi:10.1007/s006070050015.

28 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

29 Martin Klazar. The Füredi-Hajnal conjecture implies the Stanley-Wilf conjecture. In *Formal power series and algebraic combinatorics*, pages 250–255. Springer, 2000.

30 Dietrich Kuske and Nicole Schweikardt. Gaifman normal forms for counting extensions of first-order logic. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 133:1–133:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.133.

31 Bingkai Lin. The parameterized complexity of the k-biclique problem. *J. ACM*, 65(5):34:1–34:23, 2018. doi:10.1145/3212622.

32 Joseph H. Maclagan-Wedderburn. A theorem on finite algebras. *Transactions of the American Mathematical Society*, 6(3):349–352, 1905.

33 Adam Marcus and Gábor Tardos. Excluded permutation matrices and the Stanley-Wilf conjecture. *J. Comb. Theory, Ser. A*, 107(1):153–160, 2004. doi:10.1016/j.jcta.2004.04.002.

34 Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001. URL: http://www.cacr.math.uwaterloo.ca/hac/.

35 Clément Pernet. Computing with quasiseparable matrices. In Sergei A. Abramov, Eugene V. Zima, and Xiao-Shan Gao, editors, *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2016, Waterloo, ON, Canada, July 19-22, 2016*, pages 389–396. ACM, 2016. doi:10.1145/2930889.2930915.

36 Clément Pernet and Arne Storjohann. Time and space efficient generators for quasiseparable matrices. *J. Symb. Comput.*, 85:224–246, 2018. doi:10.1016/j.jsc.2017.07.010.

37 Michal Pilipczuk, Marek Sokolowski, and Anna Zych-Pawlewicz. Compact representation for matrices of bounded twin-width. In Petra Berenbrink and Benjamin Monmege, editors, *39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022, March 15-18, 2022, Marseille, France (Virtual Conference)*, volume 219 of *LIPIcs*, pages 52:1–52:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.STACS.2022.52.

38 Neil Robertson and Paul D. Seymour. Graph minors. I. Excluding a forest. *J. Comb. Theory, Ser. B*, 35(1):39–61, 1983. doi:10.1016/0095-8956(83)90079-5.

39 Neil Robertson and Paul D. Seymour. Graph minors. V. Excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41(1):92–114, 1986. URL: https://www.sciencedirect.com/science/article/pii/0095895686900304, doi:https://doi.org/10.1016/0095-8956(86)90030-4.

40 Christopher De Sa, Albert Gu, Rohan Puttagunta, Christopher Ré, and Atri Rudra. A two-pronged progress in structured dense matrix vector multiplication. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1060–1079. SIAM, 2018. doi:10.1137/1.9781611975031.69.

41 Raf Vandebril, Marc Van Barel, Gene Golub, and Nicola Mastronardi. A bibliography on semiseparable matrices. *Calcolo*, 42(3):249–270, 2005.

42 Jianlin Xia, Shivkumar Chandrasekaran, Ming Gu, and Xiaoye S. Li. Fast algorithms for hierarchically semiseparable matrices. *Numer. Linear Algebra Appl.*, 17(6):953–976, 2010. doi:10.1002/nla.691.